Automated Unbounded Verification of Stateful Cryptographic Protocols with Exclusive OR

Jannik Dreier
Université de Lorraine
CNRS, Inria, LORIA
F-54000 Nancy
France
jannik.dreier@loria.fr

Lucca Hirschi
Department of
Computer Science
ETH Zurich
Switzerland
lucca.hirschi@inf.ethz.ch

Saša Radomirović
School of
Science and Engineering
University of Dundee
UK
s.radomirovic@dundee.ac.uk

Ralf Sasse
Department of
Computer Science
ETH Zurich
Switzerland
ralf.sasse@inf.ethz.ch

***Abstract*—Exclusive-or (XOR) operations are common in cryptographic protocols, in particular in RFID protocols and electronic payment protocols. Although there are numerous applications, due to the inherent complexity of faithful models of XOR, there is only limited tool support for the verification of cryptographic protocols using XOR.**

The TAMARIN prover is a state-of-the-art verification tool for cryptographic protocols in the symbolic model. In this paper, we improve the underlying theory and the tool to deal with an equational theory modeling XOR operations. The XOR theory can be freely combined with all equational theories previously supported, including user-defined equational theories. This makes TAMARIN the first tool to support simultaneously this large set of equational theories, protocols with global mutable state, an unbounded number of sessions, and complex security properties including observational equivalence. We demonstrate the effectiveness of our approach by analyzing several protocols that rely on XOR, in particular multiple RFID-protocols, where we can identify attacks as well as provide proofs.

I. INTRODUCTION

Security protocols aim to protect communication in the presence of malicious parties, for example against an attacker on the internet or an attacker on the local wireless network. They are vital for applications including e-commerce, online voting, e-government, and online banking to ensure security properties, such as confidentiality, entity and message authentication, anonymity, and untraceability. To this end, security protocols employ cryptographic primitives, most commonly symmetric and asymmetric encryption, digital signatures, and cryptographic hash functions. Many protocols, in particular those designed for applications where the participants have limited power or computational resources, employ exclusive-or (XOR) operations. Typical examples are RFID protocols [29], [45], [42] and mobile communication protocols (3G [1], 4G/LTE [3], 5G [2]).

As illustrated by many attacks, e.g., [16], [8], [4], security protocols are notoriously difficult to get right. Consequently, many tools for the automated analysis of security protocols have been developed, e.g., [7], [21], [37], [31], [39], [48]. Historically, the main focus of these tools has been the analysis of authentication and confidentiality properties of protocols that employ standard cryptographic primitives in a Dolev-Yao adversary model. More recently there has been active research on widening the scope of automated protocol verification by extending the class of properties that can be verified to include, e.g., equivalence properties [19], [24], [27], [51], [14], extending the adversary model with complex forms of compromise [13], or extending the expressiveness of protocols, e.g., by allowing different sessions to update a global, mutable state [6], [43].

Perhaps most significant is the support for user-specified equational theories allowing for the modeling of particular cryptographic primitives [21], [37], [48], [24], [35]. However, these tools do not allow a faithful modeling of XOR or have other strong limitations such as bounded verification or no support for state. We refer to the related work section for a detailed discussion. The difficulty in faithfully modeling XOR is is due to the unique combination of algebraic properties that XOR satisfies, specifically (i) the associativity ($x \oplus (y \oplus z) = (x \oplus y) \oplus z$) and commutativity ($x \oplus y = y \oplus x$) properties and (ii) the cancellation properties (e.g., $x \oplus x = 0$). Modeling XOR without one of the two properties provides only weak security guarantees since whole classes of attacks are missed.

The TAMARIN prover [48], [56] is a state-of-the-art cryptographic protocol verifier which allows the user to combine all of the following: specifying complex security properties (both trace and equivalence properties [14]), modeling cryptographic primitives by means of equational theories, and allowing protocols to maintain state information. The class of user-specified equational theories supported by the tool is the class of convergent equational theories that have the finite variant property [30], in addition to built-in theories for Diffie-Hellman exponentiations, bilinear pairings, and multisets. Although TAMARIN supports such a large class of user-defined equational theories, one cannot simply model XOR using a user-defined theory due to its associativity and commutativity (AC) properties. At the same time, TAMARIN supports AC symbols in some built-in theories such as Diffie-Hellman exponentiation, but the cancellation properties of XOR (see (ii) above) require a special treatment, as a naive implementation often results in non-termination.

Our contributions. In this paper, we significantly extend TAMARIN to support a precise modeling of XOR operations

taking into account its AC and cancellation properties. More technically, we model XOR in TAMARIN's term algebra by introducing a symbol for XOR, treated modulo AC, that satisfies the expected cancellation properties. To avoid systematic non-termination issues in TAMARIN's backward search, the exploration needs to be constrained without affecting completeness (i.e., preserving all attacks). We show that the previously implemented constraints are not adequate for XOR since they yield systematic non-termination in the presence of XOR. We carefully devise new dedicated constraints for XOR and prove that completeness is still ensured under the combination of all constraints. Finally, we show that when analyzing equivalence properties, the previous attacker model of TAMARIN is not suitable for XOR as it yields spurious attacks (i.e., attacks that are solely artifacts of this internal model). We improve the attacker model to considerably enhance the precision of the analysis in the presence of XOR.

We have implemented these extensions in the TAMARIN prover and demonstrate that the tool succeeds to effectively analyze diverse protocols including some that were previously out of scope of automated verification.

We model and analyze in particular the RFID protocol LAK'06 [45], which consists of two roles exchanging five messages. This protocol is stateful, heavily relies on XOR, and features an else branch. We analyze secrecy, non-injective agreement (in both directions) as well as three unlinkability notions. Considering an unbounded number of sessions, we obtain semi-automatic proofs and attacks (i.e, limited number of interactions to guide state exploration, all steps machine-checked) for the reachability properties (i.e., secrecy and agreement). We obtain fully automatic proofs for and attacks on the different unlinkability notions for a bounded number of sessions. Considering such a faithful modeling, all these analyses were out of the scope of existing tools.

We fully automatically analyze two further XOR-based RFID protocols, CH'07 [29] and KCL'07 [42] against similar properties. Furthermore we study a version of Needham-Schroeder-Lowe (NSL) with XOR and a challenge response protocol using XOR. We also analyze the off-line variant of Chaum's digital cash protocol [25] which uses XOR and blind signatures. In this protocol we prove that a customer remains anonymous when not double-spending coins, and find an attack on anonymity when double-spending. Previous analyses [36], [35] could not model XOR precisely and therefore provide weaker guarantees.

Related work. In the *computational model*, XOR operations are common and supported by many tools, e.g., EasyCrypt [11] or CryptoVerif [18]. However, computational tools typically have a lesser degree of automation (e.g., EasyCrypt is mainly interactive), or are tailored to specific applications (e.g., [10], [12]).

In the *symbolic model*, there are numerous verification tools for cryptographic protocols, some of which support XOR operations.

In the case of a *bounded* number of sessions, AKISS [9] and some solvers from the AVISPA [7] suite (CL-ATSE [57] and OFMC [15]) support XOR operations. AKISS is designed to verify observational equivalence properties, while OFMC and CL-ATSE both are limited to trace properties. These tools provide weaker guarantees than TAMARIN, as they only consider a bounded number of sessions.

We will now discuss and compare our extension of TAMARIN with other tools for automated verification for an *unbounded* number of sessions.

SCYTHER [31] is restricted to a fixed set of cryptographic primitives and does not allow XOR operations. Moreover, it neither supports global mutable state nor verification of equivalence properties.

CPSA [39] was designed for analyzing, essentially, authentication and secrecy properties. The tool was used, in combination with the theorem prover PVS, to analyze stateful protocols [50]. However, like Scyther, it does neither support XOR nor the verification of equivalence properties.

MAUDE-NPA [37] offers support for many equational theories, in particular XOR. MAUDE-NPA treats algebraic properties, such as associative-commutative operators, in a more generic way than TAMARIN, which only offers support for built-in Diffie-Hellman and bilinear pairing theories, as well as multisets, and with the extension presented in this paper XOR operators. However, MAUDE-NPA does not support global mutable state nor protocols with else branches. Moreover, the verification of equivalence properties suffers from termination problems [51].

PROVERIF [21] supports user-defined equational theories, and allows for the verification of a rich variety of security properties. Moreover, the abstractions (based on a translation of applied pi calculus processes into Horn clauses) underlying the theory of PROVERIF make it very efficient. However, these abstractions may also cause false attacks, which make the tool unsuitable to analyze stateful protocols. An extension of PROVERIF (i.e., STATVERIF [6]) tries to overcome this shortcoming. However, the support for stateful protocols that can be effectively analyzed by STATVERIF remains partial. For instance, only a fixed number of state cells may be declared and non-termination arises frequently. Moreover, only secrecy properties can be verified with STATVERIF. The user-defined equational theories PROVERIF can handle are insufficient to model XOR, but there is a different extension for PROVERIF to handle some theories including XOR [44]. However, this extension is limited to secrecy and simple authentication properties, and again unsuitable for stateful protocols.

The TAMARIN front-end SAPIC [43] has been successfully used to analyze stateful protocols given in an applied pi calculus extension. It directly benefits from our work.

Although there cannot be a computational soundness proof for symbolic models of XOR [58], we argue that symbolic analysis is still useful in itself due to the higher grade of automation. This higher grade of automation allows the handling of more complex protocols, which are difficult to handle manually, and which are the goal of our work. Moreover, symbolic attacks still constitute valid attacks.

Outline. We present necessary preliminaries in Section II. Our extensions of the theory and tool are described in Section III. We evaluate the latter with the case studies shown in Section IV, and we also argue why one needs both AC and cancellation properties for faithfully modeling XOR. We give concluding remarks in Section V.

II. Preliminaries

In Tamarin, messages are represented as terms. Protocols and adversaries are modeled using multiset rewriting rules. We also show how security properties are specified.

A. Messages represented as terms

As usual in symbolic security protocol verification we represent messages and operations on them as terms in an order-sorted term algebra, using an equational theory. We assume a set of operators with their arities as signature Σ_{Op}. We define three sorts, a top sort *msg* including all terms together with two incomparable subsorts *fr* and *pub*, where terms of the former model random ("fresh") values in general, and nonces and keys in particular, while terms of the latter represent publicly known ("public") values. We also assume countably infinite sets of variables $\mathcal{V}_s$, for each sort s, and call the union of all such sets $\mathcal{V}$. We similarly treat names, with a countable set $\mathcal{N}_s$ for each sort s, and their union $\mathcal{N}$. The set of terms given by the closure of using operators from the signature containing variables in $\mathcal{V}$ and names in $\mathcal{N}$ is denoted $T_{\Sigma_{Op}}(\mathcal{V},\mathcal{N})$. A term t is *ground* if it contains no variables and we write $T_{\Sigma_{Op}}(\mathcal{N})$ for the set of all ground terms, or simply $T_{\Sigma_{Op}}$. A *substitution* σ is a function from variables to terms. We homomorphically lift substitutions to terms and use postfix notation, so $\sigma(t)$ is written $t\sigma$.

Given a signature Σ_{Op}, an *equation* is an unordered pair of terms s and t, written $s = t$, for $s,t \in T_{\Sigma_{Op}}(\mathcal{V})$. An *equational presentation* over Σ_{Op} for a given set of equations E is $\mathcal{E} = (\Sigma_{Op}, E)$. The associated *equational theory* is the smallest congruence closure containing all instances of E, for which we write $=_{\mathcal{E}}$. Whenever it is clear from the context we drop the signature and simply write $=_E$. Two terms s and t are equal modulo E if and only if $s =_E t$. We consider sets, sequences, and multisets modulo E by using the subscript E.

In Tamarin, user-defined equational theories are given using their rewrite rules oriented left to right. They have to be confluent and terminating, i.e., convergent. In this case there are unique normal forms for all terms, written $t{\downarrow_E}$. Thus we reason about terms in normal form in the following.

Example 1. To model asymmetric signatures, let Σ_{Op} be the signature consisting of the functions $sign(\cdot,\cdot)$, $checksign(\cdot,\cdot)$, and $pk(\cdot)$ together with the equation $checksign(sign(x,k),pk(k)) = x$.

Tamarin also requires that equational theories have the *finite variant property* (FVP) [30]. For a theory with the FVP, for any term t, we can compute a finite set of terms $t_1,\ldots,t_n$ with the following property: For any substitution σ, there is an $i \in \{1,\ldots,n\}$ and substitution ϕ such that $t\sigma{\downarrow_E} = t_i\phi$. This enables efficient symbolic protocol analysis by using a pre-computation to replace the equational theory. More precisely, the complete set of variants modulo E for a term t is denoted $\lceil t \rceil^E$. This set can be computed via folding variant narrowing [38].

$$x \oplus x = 0 \quad (1) \qquad x \oplus (y \oplus z) = (x \oplus y) \oplus z \quad (4)$$
$$x \oplus 0 = x \quad (2) \qquad x \oplus y = y \oplus x \quad (5)$$
$$x \oplus x \oplus y = y \quad (3)$$

Fig. 1: Equational theory E_{XOR} for XOR

XOR operations are usually (see, e.g., [30]) modeled using the equations given in Figure 1. Equation (1) models the main cancellation property of XOR, (2) models the fact that 0 is the neutral element, and (3) is required for technical reasons (to achieve AC-coherence, shown in [38]). Equations (1) to (3) can be ordered from left to right and result in a convergent rewriting system with the finite variant property, and could thus even be used in a user-specified equational theory in Tamarin. However, equations (4) and (5) cannot be handled in the same way as the resulting equational theory would not be terminating.

B. Modeling protocols and adversaries using multiset rewriting rules

Security protocols are modeled by *multiset rewriting rules*. These rules work on sets of *facts*. Formally, we assume an unsorted fact signature Σ_{Fact}, partitioned into linear and persistent facts. The set of facts is defined as $\mathcal{F} = \{F(t_1,\ldots,t_n) \mid t_i \in T_{\Sigma_{Op}}(\mathcal{V},\mathcal{N}), F \in \Sigma_{Fact} \text{ of arity } n\}$. Linear facts can only be consumed as often as they have been created, while persistent facts can be consumed an unbounded number of times. We denote the set of multisets of facts as $\mathcal{F}^\sharp$ and the set of ground facts as $\mathcal{G}^\sharp$. The function $set(\cdot)$ converts multisets to sets by dropping the multiplicity. We use the superscript $^\sharp$ to denote operations on multisets, e.g., $\cup^\sharp$ denotes the union on multisets.

Labeled multiset rewriting rules give the system's state transitions. Such rules are given as (id, l, a, r) with id a unique identifier, and l, a, and r multisets of facts. The resulting rule is $ri = id : l -[\ a\ ]\to r$. We say its *name* is $name(ri) = id$, its *premises* are $prems(ri) = l$, its *conclusions* $concs(ri) = r$, and its *actions* $acts(ri) = a$.

We denote the ground instances $ginsts(R)$ for a set of multiset rewriting rules R. We denote by $lfacts(l)$ the multiset of linear facts and by $pfacts(l)$ the set of persistent facts in l. The semantics of a set of multiset rewriting rules R are given by a *labeled transition relation* $\to_R \subseteq \mathcal{G}^\sharp \times \mathcal{G}^\sharp \times \mathcal{G}^\sharp$, defined by the following *step rule*, where S is the current state (a multiset of facts):

$$\frac{ri = id : l -[\ a\ ]\to r \in_E ginsts(R) \quad lfacts(l) \subseteq^\sharp S \quad pfacts(l) \subseteq S}{S \xrightarrow{set(a)}_R ((S \setminus^\sharp lfacts(l)) \cup^\sharp r)}$$

Note that the initial state of a labeled transition system derived from multiset rewriting rules is the empty multiset of facts $\varnothing$. The transition according to the step rule transforms a multiset of facts into another multiset of facts. The actions of the rule are the label attached to the transition. We define our security properties over these labels. We rewrite modulo equations E, so we use $\in_E$ for the rule instance modulo. Linear facts are consumed and need to be available sufficiently often, while persistent facts just need to be present. The next state is derived by removing consumed linear facts and adding all generated linear (with correct multiplicity) and persistent facts.

There is a single distinguished *fresh* rule: $\mathit{Fresh} : \multimap[]\to \mathsf{Fr}(n)$. This rule has no premise and is the only rule allowed to create the linear Fr facts. To ensure that the generated n is unique, there is an additional condition enforcing that within an execution the variables n from two instances are different. A detailed explanation is available in [54].

We define an *execution* e of a protocol P as the alternating sequence of states and rule instances: $S_0, (l_1 \multimap[\ a_1\ ]\to r_1), S_1, \ldots, S_{n-1}, (l_n \multimap[\ a_n\ ]\to r_n), S_n$ such that $S_0 = \varnothing$, and that for all $i \in \{1, \ldots, n\}$ we have all $(S_{i-1}, (l_i \multimap[\ a_i\ ]\to r_i), S_i)$ are valid steps according to the step rule. We associate the *trace* $trace(e) = [set(a_1), \ldots, set(a_n)]$ with such an execution e. We write $exec(P)$ for the set of executions of P.

We consider a standard Dolev-Yao style adversary with full control over the network and the ability to apply all operators. The message deduction rules are given by *MD* below. The adversary learns all messages sent by participants as they are put into the linear Out facts and are subsequently stored in the persistent adversary knowledge K. The adversary can send messages to protocol participants by putting them into linear In facts. Moreover, the adversary can generate his own random values and knows all public values. He can also apply functions from the signature using the rules in the third line of *MD*.

$$\begin{aligned} \mathit{MD} = \{\ & \mathsf{Out}(x) \multimap[]\to \mathsf{K}(x),\ \mathsf{K}(x) \multimap[\ \mathsf{K}(x)\ ]\to \mathsf{In}(x), \\ & \mathsf{Fr}(x{:}\mathit{fr}) \multimap[]\to \mathsf{K}(x{:}\mathit{fr}),\ []\multimap[]\to \mathsf{K}(x{:}\mathit{pub})\ \} \\ \cup\ \{\ & \mathsf{K}(x_1), \ldots, \mathsf{K}(x_n) \multimap[]\to \mathsf{K}(f(x_1, \ldots, x_n)) \\ & \mid f \in \Sigma_{Op} \text{ with arity } n\ \} \end{aligned}$$

We consider all terms modulo the given equational theory, so these rules do not deal explicitly with the equations modeling the cryptographic theory. As a more efficient form, TAMARIN uses *dependency graphs* to represent the protocol and adversary deduction rules which are applied.

Definition 1 (Dependency Graph)**.** We say that the pair $dg = (I, D)$ is a dependency graph modulo $\mathcal{E}$ for R if I is a sequence of $\mathcal{E}$-ground instances of rules from $R \cup Fresh$, $D \in \mathbb{N}^2 \times \mathbb{N}^2$, and dg satisfies the conditions:

DG1 For every edge $(i, u) \nrightarrow (j, v) \in D$ it holds that $i \leq j$ and the conclusion fact of (i, u) is syntactically equal to the premise fact of (j, v).

DG2 Every premise of dg has exactly one incoming edge.

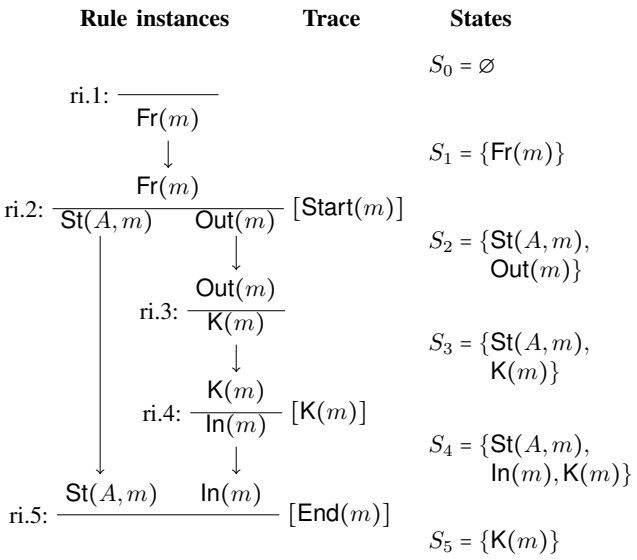

Fig. 2: Example execution of $(P_{basic} \cup \mathit{MD})$.

DG3 Every linear conclusion of dg has at most one outgoing edge.

DG4 The Fresh instances are unique.

We denote the set of all dependency graphs modulo $\mathcal{E}$ for R by $dgraphs_{\mathcal{E}}(R)$.

Example 2. Consider a protocol P_{basic} where agent A sends a nonce m on the network and then receives it, specified using the following rules:

$$P_{basic} = \left\{ \frac{\mathsf{Fr}(m)}{\mathsf{St}(A,m)\quad \mathsf{Out}(m)}[\mathsf{Start}(m)], \frac{\mathsf{St}(A,m)\quad \mathsf{In}(m)}{}[\mathsf{End}(m)] \right\}$$

Figure 2 gives a sample execution of this protocol as a dependency graph. It also illustrates how the dependency graph represents the trace and intermediate states.

We write $dgraphs(R)$ to denote all possible dependency graphs for the multiset rules R, and $traces(dg)$ for the traces of a dependency graph dg. It is easy to see that the dependency graphs have the same traces as the labeled transition system executions.

Lemma 1 ([54], Lemma 4)**.** *Let E be an equational theory. Then for all protocols P:*

$$\{trace(e) | e \in exec(P \cup \mathit{MD})\} =_E \{t | t \in traces(dg), dg \in dgraphs_E(P \cup \mathit{MD})\}$$

C. Security property specification

We consider both trace properties and indistinguishability properties. Examples of trace properties are secrecy and mutual agreement. These properties are expressed as first-order

logic formulas. Formulas use variables of a new sort, *temp*, for reasoning about the order of events, and are evaluated on traces. The informal semantics of our atomic formulas are

- $\bot$: false;
- $t_1 \approx t_2$: t_1 and t_2 are equal in the equational theory;
- $F@i$: fact $F \in_E tr[i]$ where i is of sort *temp* and $tr[i]$ is the i-th element of the trace tr on which we evaluate the formula;
- $i \doteq j$: timepoints i and j are equal;
- $i \lessdot j$: timepoint i occurs before timepoint j.

See [54] for all the details on the semantics and fragment of first order logic accepted by TAMARIN. We write $tr \vDash \varphi$ when φ holds on trace tr and lift the semantics to sets of traces: given a set of traces Tr we write $Tr \vDash^{\forall} \varphi$ if $tr \vDash \varphi$ for any $tr \in Tr$ and $Tr \vDash^{\exists} \varphi$ if $tr \vDash \varphi$ for some $tr \in Tr$.

We specify unlinkability, anonymity, and more generally equivalence properties by use of *diff*-terms (defining bi-systems, i.e., two systems differing only in some terms) and check their observational equivalence, see [14].

Example 3 ([14], Ex. 10)**.** An equational theory representing probabilistic encryption is $pdec(penc(m, pk(k), r), k) = m$. This equation gives rise to the *decryption rule* for probabilistic encryption for the adversary automatically generated by TAMARIN:

$$Dpenc : \mathsf{K}(penc(m, pk(k), r)), \mathsf{K}(k) \multimap[]\to \mathsf{K}(m)\,.$$

Consider now the following bi-system:

$$\begin{aligned} S = \{\quad GEN :&\quad \mathsf{Fr}(k) \multimap[]\to \mathsf{Key}(k), \mathsf{Out}(pk(k)),\\ ENC :&\quad \mathsf{Key}(k), \mathsf{Fr}(r_1), \mathsf{Fr}(r_2), \mathsf{In}(x) \multimap[]\to\\ &\quad \mathsf{Out}(\mathit{diff}[r_1, penc(x, pk(k), r_2)]) \qquad \}\,. \end{aligned}$$

Here TAMARIN will compare the system where $\mathit{diff}[r_1, penc(x, pk(k), r_2)]$ is replaced by r_1 to the system where it is replaced by $penc(x, pk(k), r_2)$. If the adversary cannot distinguish the two systems, they are said to be observationally equivalent. In this example, this means that he cannot distinguish a probabilistic encryption from random.

For both types of properties, TAMARIN analyzes the reachability of attack states, which are defined by a given security property. It does so by exploring symbolic traces in a backward fashion.

III. HANDLING XOR

In the following, we explain how we deal with the AC properties of the XOR theory, and how we integrate it with the existing built-in and user-defined equational theories. Moreover, we show that the existing normal form conditions are insufficient. To address this, we propose a new normal form condition that eliminates redundant steps, and show its soundness. Finally we explain why we also had to adapt the adversary model for equivalence properties.

As explained above, one cannot handle the equational theory for exclusive-or, E_{XOR}, within the user-defined theories of TAMARIN due to the combination of its associativity and commutativity (AC) properties with its cancellation properties. To deal with this equational theory, we split it into two parts: (i) the convergent equations modeling the cancellation properties, and (ii) the axioms for associativity and commutativity, and then reason modulo the AC axioms. So, we define (i) *XOR* to be the equational theory consisting of equations (1)-(3) oriented left to right, and (ii) *AC* to be the equational theory consisting of equations (4) and (5). As *XOR* has the required properties (*AC*-convergence and *AC*-coherence, see [38] for details and how *XOR* has these properties), we can define $t{\downarrow_{XOR}}$ as the normal form of term t with respect to *XOR*, *AC*-rewriting and have $s =_{E_{XOR}} t$ iff $s{\downarrow_{XOR}} =_{AC} t{\downarrow_{XOR}}$. We say that t is $\downarrow_{XOR}$-normal iff $t =_{AC} t{\downarrow_{XOR}}$.

TAMARIN already supports user-defined equational theories (without AC operators) and some built-in equational theories (with AC operators) such as the multiplication operation in the Diffie-Hellman equational theory [54], as well as bilinear pairing and multisets. To integrate the existing theories with XOR, we refer by *DHBPM* to the rewriting part of equational theory E_{DHBPM} for Diffie-Hellman exponentiation, bilinear pairing, and multisets as well as the user-defined convergent theory with the finite variant property as used in TAMARIN, and let *ACC* denote the underlying equational axioms of associativity and commutativity for multiplication, bilinear pairing, and multisets. Note that E_{DHBPM}, in particular the user-defined part, is not allowed to use the function symbols from E_{XOR}, so that the equational theories are disjoint. We now consider *ALL* = *XOR* ∪ *DHBPM*, *AC'* = *AC* ∪ *ACC* and $E_{ALL} = E_{XOR} \cup E_{DHBPM}$, i.e., the union of E_{XOR} and the existing built-in equational theories in TAMARIN.

Note that this is compatible with any user-defined equational theory as well. In a first step to enable automated analysis, we now switch from dependency graphs to dependency graphs *modulo AC* using the finite variant property. For a protocol P, we denote the variants of all protocol rules induced by the equational theory $\lceil P \rceil^E$. The following lemma establishes a strong connection between both types of dependency graphs.

Lemma 2. *For all protocols P,*

$$\begin{aligned} &dgraphs_{E_{ALL}}(P \cup MD){\downarrow_{ALL}} =_{AC'}\\ &\{dg \mid dg \in dgraphs_{AC'}(\lceil P \cup MD \rceil^{E_{ALL}}) \wedge dg \downarrow_{ALL} - normal\} \end{aligned}$$

Proof. By extension of Lemma 5 in [54]. The proof is analogous as the equational theory has the same properties. □

In a second step, we switch from dependency graphs modulo AC to *normal dependency graphs* next.

A. Normal dependency graphs

Even for simple convergent theories containing only the pairing function $\langle \cdot, \cdot \rangle$ and the *fst* and *snd* operators, non-normalized dependency graphs are not sufficient to automate the analysis of traces. For example, consider the case where the adversary deduces the first element a of a pair $\langle a, b \rangle$ by applying the function fst(·), then pairs it with an element c, and then again deduces a, this time from the new pair, to next

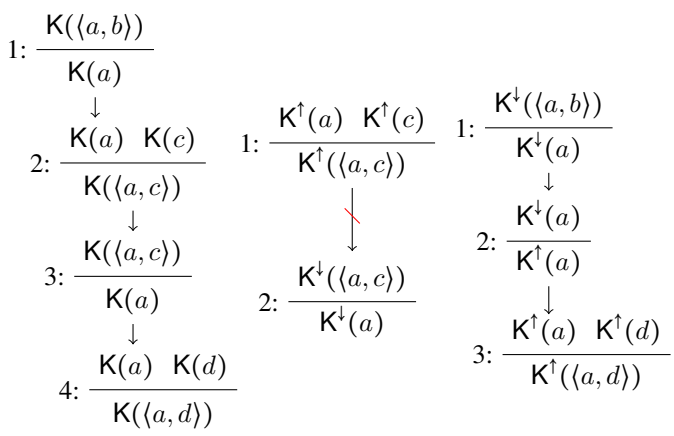

Fig. 3: Message deduction graphs for pairing: the left represents a redundant dependency graph, the middle an impossible deduction with ordered K-facts, and the right shows a shorter deduction with final conclusion equivalent to the left.

build the pair $\langle a, d\rangle$. This is visualized in the left-most graph of Figure 3. (Note that the topmost rule is actually an instance of the function application rule for fst$(\cdot)$ where the conclusion fst$(\langle a, b\rangle)$ is presented in the reduced form a, according to the equational theory.) This is a correct dependency graph, but redundant, as the steps containing c could have been skipped. As this can be resolved in just one step TAMARIN uses *normal* dependency graphs that exclude such useless steps. This is necessary as otherwise automated analysis will easily loop.

Construction and Deconstruction Rules. To improve efficiency and avoid the aforementioned redundancy, TAMARIN makes the equational theory explicit by dividing the adversary rules into two categories: construction rules and deconstruction rules. Deconstruction rules correspond to equations and are used by the adversary just after protocol rules to deduce messages from what has been sent on the network. Construction rules are, conversely, used to build messages from the adversary's knowledge that are then sent on the network. To achieve this, adversary knowledge K facts are equipped with an orientation, *up* and *down*, denoted $\mathsf{K}^{\uparrow}$ and $\mathsf{K}^{\downarrow}$. Deconstruction rules have premises with both $\mathsf{K}^{\downarrow}$ and $\mathsf{K}^{\uparrow}$ facts (as, e.g., decrypting a ciphertext that was received requires knowing the key) and a conclusion with a $\mathsf{K}^{\downarrow}$ fact. Construction rules, conversely, have premises with only $\mathsf{K}^{\uparrow}$ facts and their conclusion is a $\mathsf{K}^{\uparrow}$ fact as well. To match the purpose of construction and deconstruction rules, the new Out rule has a $\mathsf{K}^{\downarrow}$ fact as conclusion, while the In rule has $\mathsf{K}^{\uparrow}$ facts as premise. The transition from $\mathsf{K}^{\downarrow}$ to $\mathsf{K}^{\uparrow}$ is achieved by a special rule with label "Coerce", see below, but no direct conversion from $\mathsf{K}^{\uparrow}$ to $\mathsf{K}^{\downarrow}$ is possible to prevent loops. This enforces deconstruction rules to be used before construction rules.

In the case of XOR, we have two deconstruction rules and one construction rule, which directly result from the variants of $x_1 \oplus x_2$. This results in normal deduction rules depicted in Figure 4, including the usual pairing and unpairing operators.

With such rules, the adversary avoids cases of redundancy

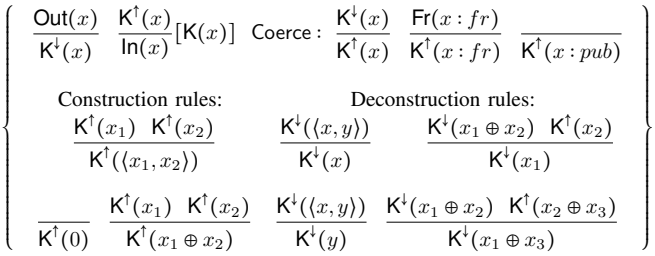

Fig. 4: Set of normal deduction rules ND_{XOR}

as shown in Figure 3. In the following we consider the normal deduction rules ND, which include ND_{XOR}, as well as the construction and deconstruction rules for Diffie-Hellman exponentiation, bilinear pairing, multisets, as well as the user-defined equational theory (see [54], [35] for details on these rules).

Normal Form Conditions. We integrate the concept of normal message deduction with construction and deconstruction rules and dependency graphs, yielding *normal dependency graphs*. Normal dependency graphs enforce additional normal form conditions, called **N1**-**N12** ([35], recalled in Appendix A). These conditions help in avoiding other redundancies by, for example, prohibiting the adversary from deducing the same term multiple times.

The existing normal form conditions are however insufficient to handle XOR, as illustrated by the following example.

Example 4. TAMARIN uses backwards constraint-solving to find normal dependency graphs representing the protocol executions. Suppose that during the constraint solving the adversary needs to compute a term $a \oplus b$, or more precisely TAMARIN encounters an unsolved $\mathsf{K}^{\downarrow}(a \oplus b)$ premise. Then TAMARIN will check all possible ways for the adversary to compute this term: The premise can either be the result of a protocol output, or the result of a deconstruction rule for XOR. In the latter case, $\mathsf{K}^{\downarrow}(a \oplus b)$ can, for example, be the conclusion of a rule instance with premises $\mathsf{K}^{\downarrow}(a \oplus c)$ and $\mathsf{K}^{\uparrow}(c \oplus b)$. TAMARIN will then try to resolve the new premises, and again $\mathsf{K}^{\downarrow}(a \oplus c)$ can be the conclusion of a deconstruction rule with premises $\mathsf{K}^{\downarrow}(a \oplus d)$ and $\mathsf{K}^{\uparrow}(d \oplus c)$, and so on, resulting in non-termination. This is illustrated in Figure 5. Note that this is not prevented by any of the previous normal form conditions, as they are focused on handling the previous equational theories.

Such loops would occur in many cases when analyzing even simple protocols containing XOR. To prevent them, we introduce a new normal form condition, **N13**, which enforces that there is no chain of applications of XOR deconstruction rules.

Definition 2 (N13). There is no chain of repeated instantiations of the deconstruction rules for XOR.

Intuitively, this does not limit the deducible terms for the adversary, as one can always cancel out all terms in a single step (i.e., using one deconstruction rule). This is formally

stated and proven below.

Definition 3. A normal dependency graph for a set of protocol rules P is a dependency graph dg such that $dg \in dgraphs(\lceil P \rceil^{ALL} \cup ND)$ and the conditions **N1**-**N13** are satisfied. We denote the set of all normal dependency graphs for P with $ndgraphs(P)$.

Let $\overline{tr}$ denote the subsequence, called observable trace, of all actions in a trace tr that are not equal to $\varnothing$. We can now prove the main correctness theorem which states that the observable traces of the protocol are identical with the observable traces of the normal dependency graphs. This is sufficient to show soundness and completeness, as TAMARIN generates all possible normal dependency graphs using constraint solving.

Theorem 1. *For all sets P of protocol rules,*

$$\overline{trace(execs(P \cup MD))} \downarrow_{ALL} =_{AC'} \overline{trace(ndgraphs(P))}.$$

Note that by relying on the observable trace we hide the adversary's deduction steps on both sides (which differ slightly due to the normalized deduction), but ensure that security properties (defined on actions) are carried over correctly. This theorem shows that by ordering the K-facts the adversary does not lose any power, and that we can simplify the deduction using the finite variant property. The proof, given in Appendix B is an extension of the proof of Theorem 1 in [35], where we need to add additional cases for the XOR rules. Normal form condition **N13** can be ensured because of the boundedness property of XOR, which allows reaching any term's normal form in at most two steps, one to cancel all duplicate terms, and another to cancel a possibly remaining single 0 if needed (in our deconstruction rules the latter case also corresponds to a single rule application). We also show in Appendix B that **N13** is sound and complete in equivalence mode using a similar argument, and that our new constraint solving rule to ensure **N13** is correct.

B. Adversary Model for Equivalence Properties

TAMARIN verifies dependency graph equivalence, that is similar to diff-equivalence in PROVERIF [19]. This notion requires that for any dependency graph in one protocol there is a corresponding ("mirrored") dependency graph in the other protocol [14].

$$\dfrac{\dfrac{\dfrac{\vdots}{\mathsf{K}^{\downarrow}(a \oplus e) \quad \mathsf{K}^{\uparrow}(e \oplus d)}\,3:}{\mathsf{K}^{\downarrow}(a \oplus d)}}{\mathsf{K}^{\downarrow}(a \oplus d) \quad \mathsf{K}^{\uparrow}(d \oplus c)}\,2:}{\mathsf{K}^{\downarrow}(a \oplus c)}$$
$$\dfrac{\mathsf{K}^{\downarrow}(a \oplus c) \quad \mathsf{K}^{\uparrow}(c \oplus b)}{\mathsf{K}^{\downarrow}(a \oplus b)}\,1:$$

Fig. 5: Example 4: Infinite chain of deconstruction rules.

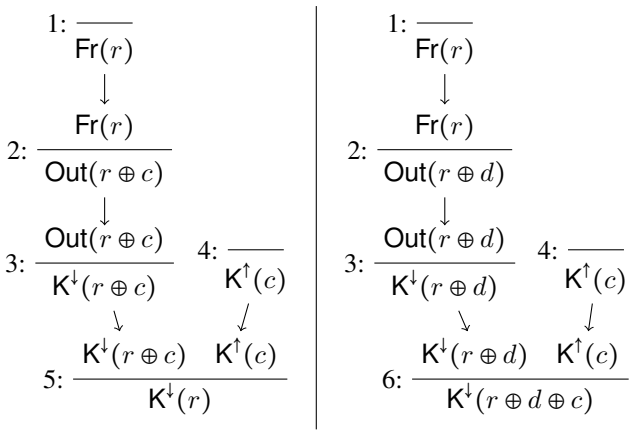

Fig. 6: Dependency graph for an instance of the example protocol and its mirror.

In doing so, TAMARIN enforces a strict one-to-one mapping of rules: an instance of a rule can only be simulated using an instance of the same rule, modulo the equational theory. For protocol rules, this means that TAMARIN allows one variant of a rule (modulo the equational theory) to be mirrored using a different variant of the same rule. For adversary rules, TAMARIN is more conservative: a deconstruction rule instance can only be simulated by another deconstruction rule instance, and not by construction rule instances, although they are technically variants of the same function application rule. This is usually desired: e.g., in the case of signatures, an instance of the deconstruction rule for the signature verification function corresponds to a successfully verified signature. If a signature can be successfully verified in one protocol, then this must also be the case in the other protocol, otherwise there is no observational equivalence.

However, in the case of XOR, this mapping is too strict, as an adversary cannot know whether an application of XOR actually canceled out some terms or not. Consider the following toy protocol with only one rule that illustrates the problem:

$$Pr : \mathsf{Fr}(r) \text{–}[\,]\text{→} \mathsf{Out}(\mathit{diff}[r \oplus c, r \oplus d])$$

where c and d are constant functions in the signature. The left and right instances of the bi-system should obviously be equivalent, as the adversary does not know the fresh value r (similar to a one-time pad encryption).

Consider now the dependency graphs given in Figure 6. The dependency graph on the left hand side corresponds to a protocol execution of the left bi-system. If we use TAMARIN's usual mirroring, this graph has no mirror, since in the right execution the rule at the bottom is not a valid instance of any deconstruction rule for XOR (cf. definition of ND_{XOR}), and TAMARIN would thus claim an attack. However, the last rule still corresponds to an application of XOR, so the adversary should not see any difference, as long as he cannot distinguish the resulting terms.

To prevent these spurious attacks, we modified TAMARIN to treat the XOR rules just like normal protocol rules, i.e., when mirroring an instance of an XOR construction or deconstruction rule, any instance corresponding to an application of the XOR operator is accepted (even if this does not strictly correspond to a standard construction or deconstruction rule as the $\uparrow$ and $\downarrow$ might be different), so long as it uses the same input and all other constraints still hold. Hence the dependency graph on the right of Figure 6 is considered a valid mirror.

Note that TAMARIN still verifies that all equalities from one side carry over to the other side, and vice versa. In particular, if we modify the protocol to also output the random value r, TAMARIN reports an attack: the adversary can cancel out r, and compare the result with the public constants c or d.

IV. CASE STUDIES

We first present a simple challenge-response protocol for illustration purposes (Section IV-A). We then present our main case study LAK'06 [45], introduce security properties, and summarize our analysis results (Section IV-B). We present our results on other RFID protocols (Sections IV-C, IV-D), an eCash protocol (Section IV-E), and a version of the Needham-Schroeder-Lowe public key protocol with XOR (Section IV-F) after that. All our TAMARIN models are freely available in the TAMARIN repository [55] folder `/examples/csf18-xor` and TAMARIN as of v1.4.0 contains the presented XOR extension. We present an overview of the results in Figure 12 in Section IV-G.

A. Introductory CR⊕ Example

We start with a toy example protocol to illustrate that a faithful model of XOR with support for AC and cancellation is necessary, as otherwise attacks can be missed. (Note that the attack we find on this example with TAMARIN is the basis of an attack on the real LAK'06 case study that follows.) Consider the following basic challenge response protocol called CR:

$$\begin{aligned} &A\text{: knows}(k)\\ &B\text{: knows}(k)\\ &A\text{: fresh}(na)\\ \textbf{CR1.}\quad &A \to B\text{: } na\\ \textbf{CR2.}\quad &B \to A\text{: } \langle h(na,k,nb), nb\rangle \end{aligned}$$

TAMARIN automatically proves both aliveness and recent-aliveness of the responder B, which is the protocol's goal.

Consider now an extension of CR that uses $\oplus$, called CR$\oplus$:

$$\begin{aligned} &A\text{: knows}(k)\\ &B\text{: knows}(k)\\ &A\text{: fresh}(na)\\ \textbf{CRx1.}\quad &A \to B\text{: } na\\ \textbf{CRx2.}\quad &B \to A\text{: } \langle h(na \oplus k \oplus nb), nb\rangle \end{aligned}$$

Our model for this protocol is bounded to two agents, and TAMARIN automatically proves aliveness of the responder B, but finds an attack on recent-aliveness, also automatically. Note that this attack is expected as an attacker can craft a correct response after having observed one real response. This attack uses the same hash value as the legitimate previous run, $h(na \oplus k \oplus nb)$, but the attacker chooses the nonce $nb' := na \oplus nb \oplus na'$ where na and nb are the values from the original run, and na' is the new challenge. This works because $na' \oplus k \oplus nb' = na' \oplus k \oplus na \oplus nb \oplus na' = k \oplus na \oplus nb = na \oplus k \oplus nb$ and thus A accepts the old hash, as it expects $h(na' \oplus k \oplus nb')$. This attack heavily relies on the cancellation property of XOR as seen in the above equality, where the duplicate na' is removed. Additionally, it also requires proper support for the AC property of the $\oplus$-operator. Without AC, one would have to define specific cancellation equations, e.g., $X \oplus (Y \oplus (X \oplus Z)) = Y \oplus Z$, pick nb' more carefully as $nb' := na' \oplus (na \oplus nb)$ and as result would then have $na' \oplus (k \oplus nb') = na' \oplus (k \oplus (na' \oplus (na \oplus nb)) = k \oplus (na \oplus nb)$ which at least contains the same values, but still not in the same order.

B. LAK'06

Protocol Description. LAK'06 [45] is an RFID protocol that aims at mutual authentication of a tag and reader while providing untraceability for the tag. In order to achieve untraceability, the protocol relies on a challenge-response mechanism based on a shared secret that is modified at the end of each session. We suppose that initially each tag has its own key k and the reader maintains a database containing those keys. To prevent desynchronization of the state k, the reader also stores the last successful key k_0 associated with each tag. Initially, $k^{\text{init}} = \text{h}(k_0^{\text{init}})$ where $\text{h}(\cdot)$ is some hash function. An Alice&Bob description of the protocol is depicted in Figure 7a.

The reader starts by challenging the tag (**1.**) with a fresh random r_0, and the tag's expected reply (**2.**) both (i) proves the knowledge of the secret k bound to the fresh value r_0 and (ii) challenges back the reader. Upon receiving that message, the reader is assured that the current key stored in the tag is k and thus updates its secrets: k_0 becomes the current key and k becomes the next key to be used: $\text{h}(k)$. It then replies to the tag's challenge proving it also knows the secret key k. When receiving that reply, the tag updates its key to $\text{h}(k)$.

When all messages reach their recipients, tag and reader stay synchronized, storing the same key $k = \text{h}^i(k^{\text{init}})$ where i is the number of successful sessions (the reader additionally stores $k_0 = \text{h}^{i-1}(k^{\text{init}})$). However, if message **3.** is lost or intercepted, the tag does not update while the reader has already done so. In that situation, the tag stores k and the reader stores $\text{h}(k)$ (as well as $k_0 = k$). In order to recover from such desynchronized states, LAK'06 allows the reader to accept the tag's replies based on the old key k_0. This mechanism is depicted in Figure 7b (reader accepts incoming message of the form **2'**.). Note that the reader does not update keys in order to re-synchronize with the tag.

Modeling in Tamarin. LAK'06 is a stateful protocol whose states are non-monotonic (i.e., if some tag stores k at some point, k will be replaced by another value later on). Additionally, we aim at modeling an unbounded number of sessions of the protocol in order to establish strong security guarantees. No tool other than TAMARIN can handle both features, and

R: knows$(k, k0)$
T: knows(k)
R: fresh$(r0)$
1. $R \to T$: r_0
T: fresh$(r1)$
2. $T \to R$: $r_1, h(r_0 \oplus r_1 \oplus k)$
R: updates $k_0 := k$
R: updates $k := h(k)$
3. $R \to T$: $h(h(r_0 \oplus r_1 \oplus k) \oplus k \oplus r_0)$
T: updates $k := h(k)$

(a) Core protocol

R: knows$(k, k0)$
T: knows$(k0)$
R: fresh$(r0)$
1. $R \to T$: r_0
T: fresh$(r1)$
2'. $T \to R$: $r_1, h(r_0 \oplus r_1 \oplus k_0)$
3'. $R \to T$: $h(h(r_0 \oplus r_1 \oplus k_0) \oplus k_0 \oplus r_0)$
T: updates $k := h(k)$

(b) Re-synchronization mechanism

Fig. 7: Alice&Bob description of LAK'06 [45]

we leverage its new capability of dealing with XOR theories to provide the first faithful modeling of the protocol[1] and first formal analysis of both reachability and equivalence properties.

Security properties. We are interested in analyzing secrecy, authentication, and untraceability properties.

Reachability properties. First, we would like to prove that, whenever a tag or a reader stores a key, then the attacker never learns the key (in the past or in the future). Formally, such a property is formalized in TAMARIN using the formula defined below, where facts $\mathsf{Claim\_Secret}(a, k)$ are produced for each rule of agent a (some tag or reader) which accesses or stores a key k.

Definition 4. *Secrecy* is modeled via the following formula:

$$\forall a\ t\ i.\ \mathsf{Claim\_secret}(a, t)@i \Rightarrow \neg(\exists j.\ \mathsf{K}(t)@j)$$

Next, we define the non-injective agreement [47] property. We assume that the TAMARIN model is equipped with facts $\mathsf{Claim\_commit}(a, b, \langle A, B, t\rangle)$ (i.e., an agent a of role A claims it has established agreement on data t with b whose role is B) and $\mathsf{Claim\_running}(b, a, \langle A, B, t\rangle)$ (i.e., an agent b of role B claims it tries to establish agreement on data t with a whose role is A).

Definition 5. *Non-injective agreement* on data t of a role A towards a role B is modeled via the following formula:

$$\forall a\ b\ t\ i.\ \mathsf{Claim\_commit}(a, b, \langle A, B, t\rangle)@i \Rightarrow (\exists j.\ \mathsf{Claim\_running}(b, a, \langle A, B, t\rangle)@j)$$

Behavioral equivalence properties. We are interested in analyzing untraceability which is one of the key requirements of LAK'06. There are various notions of untraceability which have been defined in the symbolic model in various frameworks (see comparisons in [23], [22]). We have chosen to analyze three notions of untraceability following three generic constructions: (symbolic) *game-based unlinkability* [23] (no relation to game-based computational proofs), which we call UK_1 in the rest of the paper, unlinkability as being checked in [9] (UK_2), and *strong unlinkability* (UK_3) following [40] (itself strengthening [4]). We informally define below those unlinkability properties as behavioral equivalences between two different systems involving tags and readers of different identities. Note that a tag and readers of same identity initially share the same key k, while agents of different identities initially have distinct keys k. We give identities to readers because we consider reader sessions that expect to interact with a tag of a specific identity.

Those constructions are unrelated in general [23], it seems that UK_3 is strictly stronger than the others [41] for realistic classes of protocols. Comparing the three notions is out the scope of this paper but being able to analyze all of them enables us to provide more fine-grained security guarantees.

Definition 6 (UK_1, informal)**.** UK_1 compares two systems organized in two phases:

- during the *learning phase*, the attacker can run one session for two tags and readers of identity id_1 and id_2 (the identity refers to the key a tag and a reader initially share). This phase is identical for the two systems being compared.
- during the *guessing phase*, the attacker can run a tag of identity id_1 (resp. id_2) in the first (resp. second) system.

The property holds when the two systems are behaviorally equivalent.

The two phases shall be understood as *weak phases* [32] (also called *stages* [17]): no action from the learning phase can be executed once an action from the guessing phase has been executed. Intuitively, if the attacker can distinguish the two systems then he must have a criterion to guess which is the unknown tag in the guessing phase (i.e., id_1 in the first system, id_2 in the second system). Here, the attacker can take advantage of having interacted with tags and readers of identity id_1 and id_2 during the learning phase.

Definition 7 (UK_2, informal)**.** UK_2 holds when the two following systems are behaviorally equivalent:

[1] [40] considers a stateless abstraction, [9] only considers a small number of sessions.

1) the first system is made of a tag of identity id_1 and a reader of identity either id_1 or id_2 (non-deterministic choice).
2) the second system is made of a tag of identity id_2 and a reader of identity either id_1 or id_2 (non-deterministic choice).

Additionally, for both systems, we assume that the attacker initially knows one full transcript of a past honest interaction between the tag and the reader of identity id_1.

Intuitively, if the attacker can distinguish the two systems then he is able to observe a link in the first system between the past transcript (whose identity is id_1) with data he can obtain from the tag id_1 while he doesn't observe such a link in the second system (which has no tag of identity id_1).

Definition 8 (UK_3, informal)**.** UK_3 holds when the two following systems are behaviorally equivalent:

1) the first system is made of a tag and a reader sharing the same identity that can perform two sessions each;
2) the second system is made of two different pairs of tag and reader that can only perform one session each.

The second system corresponds to an ideal scenario where there is nothing to link: no agent can be tracked because each agent plays at most one session. This is not the case for the first scenario where two sessions are considered for one single tag and one single reader. Intuitively, if the attacker cannot distinguish both systems then he has no way to track an agent over two sessions.

Note that for all unlinkability notions, we are considering finitely many agents and sessions only. For the case of strong unlinkability (UK_3), this is due to a known lack of precision of diff-equivalence when it comes to verifying such strong properties [40], [41]. Fortunately, for a bounded number of sessions only, one can overcome this limitation in TAMARIN using a simplified swapping approach [20]. On the contrary, this is not a theoretical limitation for UK_1 and UK_2 (which is the case for other existing tools that could handle XOR) but rather a pragmatic approach: as proofs are getting highly complex when combining behavioral equivalence, XOR reasoning, and stateful protocols, verification requires either an excessive amount of resources or some manual work. We thus prefer to analyze more case studies but for slightly weaker notions of unlinkability.

Analysis.

Analysis of reachability properties. We model secrecy of stored keys (Definition 4) as well as the two non-injective agreement properties for both sides (Definition 5), for one pair of tag and reader that can play an unbounded number of sessions each.

We devise a dedicated *oracle* for the LAK'06 protocol to encode superior proof-search choices compared to the default heuristics. Oracles offer a light-weight tactics language to guide the proof search in TAMARIN. This was necessary, as the proofs involve inductive reasoning to cope with the stateful nature of the LAK'06 protocol, unbounded number of sessions, and intricate and long message deductions for the equational theory including XOR. The oracle allows us to automatically complete very long and highly technical subproofs in order to focus the manual exploration on the high-level proof structure. We can thus semi-automatically prove secrecy of stored keys and non-injective agreement of the tag role towards the reader role and TAMARIN automatically finds an attack on non-injective agreement of the reader towards the tag. The attack corresponds to the one described in [34] and works just like the one presented for CR⊕ in Section IV-A.

Analysis of behavioral equivalence properties. We analyze $\mathrm{UK}_1, \mathrm{UK}_2$, and UK_3 using diff-equivalence verification in TAMARIN [14]. However, since we consider those properties for a bounded number of sessions only (1 or 2 sessions), we have not modeled the key update mechanism and used an abstraction where keys are chosen fresh for each session (e.g., as done in [40]). Thanks to those simplifications, we were able to obtain fully automatic proofs as described next.

For UK_2 (see Definition 7) we obtain a fully automatic proof.

We analyze UK_3 for four sessions in total, 2 sessions of tag and 2 sessions of reader (see Definition 8). TAMARIN automatically finds an attack. While it was known that the property fails to hold [40], TAMARIN finds a slightly different attack. Note that the protocol is claimed to be untraceable in [34] for a weaker notion of unlinkability. Strong unlinkability as being checked here may be considered too strong but [41], [40] discuss how a variant of the attack found here constitutes a practical privacy breach.

We are not able to directly analyze UK_1 when taking readers into account during the learning phase. Indeed, for such a model, one needs a *restriction*[2] for modeling *weak phase*, that states that no action from the learning phase can be executed after an action from the guessing phase. Such a restriction is crucial for avoiding obvious false attacks. However, when dealing with both restrictions and diff-equivalence, TAMARIN's analysis frequently does not terminate, which is a known limitation. Therefore, in order to get rid of that restriction, we limit our analysis of weak unlinkability to tags only. For that weak model, TAMARIN automatically proves that UK_1 holds, without oracle.

C. CH'07

Protocol Description. CH'07 [29] was designed to be a challenge-response RFID authentication protocol that provides tag untraceability. We base our model on [34]. Figure 8 shows the protocol description.

The reader R and tag T share secrets k and ID. The reader challenges the tag with a random bit string r_1, modeled as a nonce. T generates a nonce r_2 and replies with a term derived from $h(r_1 \oplus r_2 \oplus k)$, where h is a hash function, and the tag's identifier ID. The hash and ID are used as input for a function rot in which the bit string ID is rotated by a value depending on the hash. We model rot as a hash function, but

[2] *Restrictions* can be used in a TAMARIN model to restrict considered executions to the ones that satisfy a specific property.

R: knows(k, ID)
T: knows(k, ID)
R: fresh$(r1)$

CH-1. $R \to T$: $r1$
T: fresh$(r2)$

CH-2. $T \to R$: $r2, lh(rot(ID, h(r1 \oplus r2 \oplus k)) \oplus h(r1 \oplus r2 \oplus k))$

CH-3. $R \to T$: $rh(rot(ID, h(r1 \oplus r2 \oplus k)) \oplus h(r1 \oplus r2 \oplus k))$

Fig. 8: The CH'07 RFID authentication protocol.

note that a better approximation would be provided by the identity function. The term sent from the tag to the reader is the output of the lh function which consists of the first half of its argument bitstring. We model lh as a hash function.

The reader performs the same computation as the tag to identify the tag. If the tag can be identified, the reader replies with the output of the rh function, which is the second half of the bitstring the reader computed. We model rh as a hash function, too.

Analysis of reachability properties. We consider several authentication properties: recent aliveness of tag and reader as in [33], [34], and non-injective agreement (Definition 5) of tag and reader, on different data items, namely on $\langle k, r_1, r_2 \rangle$ and on $k \oplus r_1 \oplus r_2$.

The protocol does not satisfy recent aliveness of tags, due to an impersonation attack [33], [34]. The adversary can impersonate a tag to a reader after one interaction with a tag. We find this attack automatically using our TAMARIN model. Recent aliveness of the reader R is satisfied and we find an automatic proof.

Agreement on $\langle k, r_1, r_2 \rangle$ is not satisfied for either role. In both cases, this is because the adversary can modify both the challenge and the response by an xor with a term x. The adversary's modifications cancel out and both parties complete their protocol runs, but agreement is not satisfied on the nonces r_1 and r_2. We again find the attacks automatically using TAMARIN.

If we require agreement on the data $k \oplus r_1 \oplus r_2$ instead, then both agreement claims are satisfied and TAMARIN finds the proof automatically.

Analysis of behavioral equivalence. The protocol satisfies UK_1 and UK_2 (Definitions 6 and 7) and TAMARIN finds a proof automatically. The protocol does not satisfy the property UK_3 (Definition 8) because the same reader responds to a replayed query with the same answer when the second time the attacker picks $r_2' = r_1' \oplus r_1 \oplus r_2$, similarly to the attack in Section IV-A. This allows the attacker to distinguish the two systems defining the UK_3 property: in the first system, the attacker will receive the same response from the RFID reader in both sessions while in the second system the attacker will receive different responses. TAMARIN finds the attack automatically.

The interpretation of the attack is that it allows the adversary to test whether a reader is still accepting a given tag. If tags can expire or be removed from a system, then this attack allows the adversary to learn whether a previously observed tag has expired or whether it is still valid.

R: knows(k, ID)
T: knows(k, ID)
R: fresh$(r1)$

KCL-1. $R \to T$: $r1$
T: fresh$(r2)$

KCL-2. $T \to R$: $\langle ID \oplus r2, h(r1, k) \oplus r2 \rangle$

Fig. 9: The KCL'07 RFID authentication protocol.

D. KCL'07

KCL'07 [42] is an RFID protocol attempting to both authenticate the tag and to provide untraceability for the tag. It succeeds in providing the recent aliveness property for authentication, but untraceability does not hold. Figure 9 shows the protocol description.

Informally, the aliveness property is guaranteed due to the response including the hash of the challenge nonce $r1$ together with the key k that is only used inside the hash and thus can never be learned by an attacker. Note that the reader cannot learn the value $r2$ actually chosen by the tag, as a man-in-the-middle attacker can simply pick a random $r3$ and XOR it to both elements of the pair the tag sends back to the reader. Then, the reader believes $r2 \oplus r3$ to be the chosen random value. The reader can only check that the application of XOR to both values results in $ID \oplus h(r1, k)$. Therefore, the random value $r2$ is not relevant.

We thus apply a simplification in our model for the reader, which is that it receives just one term (the XOR of the elements of the pair output by the tag), while the tag sends the pair using $r2$. To be precise, the sent term is $\langle id \oplus r2, h(r1, k) \oplus r2 \rangle$ received as $\langle idr2, hashr2 \rangle$ while in our model the reader simply receives X, interpreted as $idr2 \oplus hashr2$.

The first property expected from this protocol is recent aliveness, which we prove automatically with TAMARIN. As explained in Deursen and Radomirović [34] in this protocol the tag is not actually untraceable which can be seen by sending the same nonce twice to a tag, and applying $\oplus$ to the elements of the response pair, and comparing the results. For the same challenge $r1$, the results of applying XOR to the response for a given tag is always $ID \oplus h(r1, k)$. No reader is necessary for this attack.

TAMARIN automatically finds attacks on UK_1 and UK_2. For UK_3, we construct the attack manually using TAMARIN.

E. Chaum's Offline eCash Protocol

Protocol Description. Chaum's offline eCash [26] is a suite of three protocols that allows for anonymous, untraceable, electronic currencies to be spent offline, i.e., without the need to contact a bank during the payment. The protocols involve three parties, the customer C, the seller S, and the bank B and consist of the withdrawal protocol, where C withdraws eCash from the bank, the payment protocol, where C pays S, and

eCash-1. $C \to B$: $\text{blind}(x \oplus C, r), \text{blind}(x, r)$
eCash-2. $B \to C$: $\text{sign}(\text{blind}(x \oplus C, r), skB)$, $\text{sign}(\text{blind}(x, r), skB)$
eCash-3. $S \to C$: z
eCash-4. $C \to S$: $\langle x', \text{sign}(x', skB)\rangle$ where $x' = x$ if $z = 0$ and $x' = x \oplus C$ otherwise
eCash-5. $S \to B$: $\langle x', \text{sign}(x', skB)\rangle$
eCash-6. $B \to S$: x'
eCash-7. $S \to B$: ok/not ok

Fig. 10: A simplified model of Chaum's offline eCash protocol.

eC-1. $C \to E$: $\text{blind}(x \oplus C, r), \text{blind}(x, r)$
eC-2. $E \to C$: $\text{sign}(\text{blind}(x', r), skB)$ where $x' = x$ or $x' = x \oplus C$
eC-3. $C \to E$: $\langle x', \text{sign}(x', skB)\rangle$

Fig. 11: The simplified model of Chaum's offline eCash protocol with dishonest seller and bank represented by the adversary-controlled environment E.

the deposit protocol, where S deposits the received currency with B.

The protocol uses XOR as well as blind signatures, to ensure anonymity. The blind signatures require a non subterm-convergent equational theory, which illustrates that TAMARIN is able to handle the combination of these two complex equational theories.

We model a simplified version of these protocols by combining them into one protocol consisting of three (weak) phases comprising withdrawal, payment, and deposit. We focus solely on the mechanism that provides anonymity as long as the customer does not double-spend a coin. We do not model the cut and choose procedure during withdrawal and instead assume that the customer generates well-formed coins.

The simplified protocol aims to provide anonymity to the customer as long as he does not double-spend a coin and is shown in Figure 10. Messages eCash-1 and eCash-2 model the withdrawal phase, the next two messages model the payment phase, and the remaining messages model the deposit (redemption) phase of the protocol.

Analysis. In our TAMARIN model, we consider seller and bank to be corrupted. This leads to a model where the customer interacts with the environment E that is controlled by the adversary and represents both S and B. The model is shown in Figure 11. We model customer anonymity as secrecy of the customer's identity C. TAMARIN automatically verifies that an honest customer is anonymous and finds an attack on anonymity when the customer double spends.

F. NSLPK3xor

This is an insecure variant of the Needham-Schroeder-Lowe [46] (NSL) 3-message public-key protocol. The difference to the classical NSL protocol is the presence of an exclusive-or $\oplus$ instead of concatenation in message NSLx2. This idea is due to Chevalier et al. [28], and this protocol has previously been analyzed by Sasse et al. [52] using MAUDE-NPA.

A: $\text{knows}(sk(A), pk(B))$
B: $\text{knows}(sk(B), pk(A))$
A: $\text{fresh}(na)$
NSLx1. $A \to B$: $\text{enc}(\langle na, A\rangle, pk(B))$
B: $\text{fresh}(nb)$
NSLx2. $B \to A$: $\text{enc}(\langle na, nb \oplus B\rangle, pk(A))$
NSLx3. $A \to B$: $\text{enc}(nb, pk(B))$

Our TAMARIN theory models the case where there is only one key per agent and TAMARIN automatically finds attacks on the secrecy of the two nonces as well as on the injective agreement property, as described in detail in [52].

G. Summary of Experimental Results

We present in Figure 12 a summary of the protocols we analyzed, the result obtained, the runtime required, as well as the level of automation achieved. All experiments are conducted on a server with 2 CPUs of type Intel(R) Xeon(R) CPU E5-2650 v4 @ 2.20GHz (with 12 cores each), 256GB of RAM, running Ubuntu 16.04.3, and we use 10 threads per experiment.

All our theories are available at [55], in the folder `/examples/csf18-xor`. Note that all theories also contain a lemma showing that the protocol is actually executable, avoiding modeling mistakes, all of which are verified, but not listed in the table. We list what modifications we made on each case study, or other modeling limitations, in the *Modifications* column. Then we give the result, either ✓ (the property is verified) or ✗ (the property does not hold - there is an attack).

For the automation level we show A for automated proofs or attacks, meaning using the standard heuristic and proof tree exploration. We show A (BFS) if the standard heuristic was used to find attacks, but using a breadth-first search of the proof tree (this is a common technique in TAMARIN, as BFS is often quicker at finding attacks, but not for proofs). We write O when an oracle was used to automatically find the results, as described before. We use SM for a semi-manually achieved proof, which means a short human exploration of the state-space was conducted, and then the remainder of the proof is automatic using the oracle. There is one case where we manually constructed an attack, denoted M.

V. CONCLUSION

We have extended the TAMARIN prover with equational theories including XOR, consequently expanding the class of protocols that can be faithfully modeled and analyzed using automatic verifiers. As TAMARIN is sound and complete, we cannot hope for guaranteed termination since the underlying problem is undecidable. However, our new normal form conditions, heuristics, and use of light-weight tactics encoded as oracles allow for a good level of automation as suggested by our numerous case studies. We believe that TAMARIN can now tackle large-scale real-life protocols with XOR.

As future work, we would like to study mobile telecommunication protocols that use XOR such as the AKA protocol [3],

Protocol Name	Property	Modifications	Result	Automation	Runtime
CR-xor	aliveness tag		✓	A	0.7s
	recent aliveness tag		✗	A	1.3s
NSL-xor	nonce secrecy		✗	A	5.1s
	injective agreement initiator		✗	A	1.4s
	injective agreement responder		✗	A	4.0s
CH07	recent aliveness tag		✗	A	2.2s
	recent aliveness reader		✓	A	1.1s
	non-inj. agree. tag $(k \oplus r1 \oplus r2)$		✓	A	1.1s
	non-inj. agree. tag $(k, r1, r2)$		✗	A	1.3s
	non-inj. agree. reader $(k \oplus r1 \oplus r2)$		✓	A	1.0s
	non-inj. agree. reader $(k, r1, r2)$		✗	A	1.1s
Chaum Offline	*coins	No phases	✓	O	10.4s
	anonymity	No phases	✓	A	2.8s
	anonymity of double spender	No phases	✗	A	12.7s
KCL07	recent aliveness tag	XORed pair	✓	A	1.7s
LAK06	*helpingSecrecy		✓	A	0.4s
	non-injective agreement tag		✓	A	22.0s
	non-injective agreement reader		✗	A	1.3s
LAK06-state	*helpingUpdateKey		✓	O	21.1s
	*helpingStackHash		✓	A	19.6s
	*helpingSecrecy		✓	SM	†31.6s
	non-injective agreement tag		✓	SM	†55.7s
	non-injective agreement reader		✗	A (BFS)	1m 38.7s
CH07	UK1		✓	A	51.5s
	UK2		✓	A	12m16.6s
	UK3		✗	A	3m07.1s
KCL07	UK1	XORed pair	✗	A (BFS)	126m27.6s
	UK2	XORed pair	✗	A	1m02.6s
	UK3	XORed pair	✗	M	†2.1s
LAK06	UK1 (tags only)	bounded sessions	✓	A	4m25.4s
	UK2	bounded sessions	✓	A	15m10.4s
	UK3	bounded sessions	✗	A	85m46.1s

Fig. 12: Summary Table of Results. Automation: A = automatic, A (BFS) = automatic with breadth-first search for attacks, O=oracle, SM=semi-manual, M=manual. † means the time is for verifying the resulting stored proof. More details in Section IV-G. Properties starting with * are intermediate helping lemmas in the modular proofs.

[1]. Previous analyses of this protocol [5], [49] were not able to model parts of the protocol making use of XOR, therefore providing guarantees that are too weak in our opinion. We would like to investigate the new perspectives opened by our extension in TAMARIN and faithfully analyze the AKA protocol as used in existing mobile networks (i.e., 3G [3] and 4G [1]) as well as in 5G that is being standardized [2]. On the theory side an interesting challenge is to lift some of the limitations of equivalence proofs in TAMARIN. For example, dealing with the combination of diff-equivalence and restrictions could enable additional case studies, including protocols using weak phases.

ACKNOWLEDGEMENTS

We are grateful for the support from the ERC under the EUs Horizon 2020 research and innovation program (grant agreements No 645865-SPOOC) and the UK Engineering and Physical Sciences Research Council (EPSRC) grant EP/P013694/1.

REFERENCES

[1] 3GPP: 3G security; Security architecture. TS 33.102, 3rd Generation Partnership Project (3GPP), http://www.3gpp.org/DynaReport/33102.htm

[2] 3GPP: Security Architecture and Procedures for 5G System. TS 33.501, 3rd Generation Partnership Project (3GPP), http://www.3gpp.org/DynaReport/33501.htm

[3] 3GPP: System Architecture Evolution (SAE). TR 33.401, 3rd Generation Partnership Project (3GPP), http://www.3gpp.org/DynaReport/33401.htm

[4] Arapinis, M., Chothia, T., Ritter, E., Ryan, M.: Analysing unlinkability and anonymity using the applied pi calculus. In: Proceedings of the IEEE Computer Security Foundations Symposium. IEEE Comp. Soc. Press (2010)

[5] Arapinis, M., Mancini, L., Ritter, E., Ryan, M., Golde, N., Redon, K., Borgaonkar, R.: New privacy issues in mobile telephony: fix and verification. In: Proceedings of the 2012 ACM conference on Computer and communications security. pp. 205–216. ACM (2012)

[6] Arapinis, M., Ritter, E., Ryan, M.: StatVerif: Verification of stateful processes. In: Proc. 24th IEEE Computer Security Foundations Symposium (CSF'11). pp. 33–47. IEEE Press (2011)

[7] Armando, A., Basin, D.A., Boichut, Y., Chevalier, Y., Compagna, L., Cuéllar, J., Drielsma, P.H., Héam, P.C., Kouchnarenko, O., Mantovani, J., Mödersheim, S., von Oheimb, D., Rusinowitch, M., Santiago, J., Turuani, M., Viganò, L., Vigneron, L.: The AVISPA tool for the automated validation of internet security protocols and applications. In: CAV. pp. 281–285 (2005)

[8] Armando, A., Carbone, R., Compagna, L., Cuellar, J., Tobarra, L.: Formal analysis of saml 2.0 web browser single sign-on: breaking the saml-based single sign-on for google apps. In: Proceedings of the 6th ACM workshop on Formal methods in security engineering. pp. 1–10. ACM (2008)

[9] Baelde, D., Delaune, S., Gazeau, I., Kremer, S.: Symbolic verification of privacy-type properties for security protocols with XOR. In: 30th IEEE Computer Security Foundations Symposium, CSF 2017. pp. 234–248. IEEE Computer Society (2017)

[10] Barthe, G., Crespo, J.M., Grégoire, B., Kunz, C., Lakhnech, Y., Schmidt, B., Béguelin, S.Z.: Fully automated analysis of padding-based encryption in the computational model. In: Sadeghi, A., Gligor, V.D., Yung, M. (eds.) 2013 ACM SIGSAC Conference on Computer and Communications Security, CCS'13, Berlin, Germany, November 4-8, 2013. pp. 1247–1260. ACM (2013)

[11] Barthe, G., Dupressoir, F., Grégoire, B., Kunz, C., Schmidt, B., Strub, P.: Easycrypt: A tutorial. In: Aldini, A., Lopez, J., Martinelli, F. (eds.) Foundations of Security Analysis and Design VII - FOSAD 2012/2013 Tutorial Lectures. Lecture Notes in Computer Science, vol. 8604, pp. 146–166. Springer (2013)

[12] Barthe, G., Grégoire, B., Schmidt, B.: Automated proofs of pairing-based cryptography. In: Ray, I., Li, N., Kruegel, C. (eds.) Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12-6, 2015. pp. 1156–1168. ACM (2015)

[13] Basin, D., Cremers, C.: Know your enemy: Compromising adversaries in protocol analysis. ACM Trans. Inf. Syst. Secur. 17(2), 7:1–7:31 (Nov 2014)

[14] Basin, D., Dreier, J., Sasse, R.: Automated symbolic proofs of observational equivalence. In: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security. pp. 1144–1155. ACM (2015)

[15] Basin, D., Mödersheim, S., Viganò, L.: Ofmc: A symbolic model checker for security protocols. International Journal of Information Security 4(3), 181–208 (Jun 2005)

[16] Bhargavan, K., Lavaud, A.D., Fournet, C., Pironti, A., Strub, P.Y.: Triple handshakes and cookie cutters: Breaking and fixing authentication over TLS. In: 2014 IEEE Symposium on Security and Privacy. pp. 98–113. IEEE (2014)

[17] Blanchet, B., Abadi, M., Fournet, C.: Automated verification of selected equivalences for security protocols. Journal of Logic and Algebraic Programming (2008)

[18] Blanchet, B.: A computationally sound mechanized prover for security protocols. In: 2006 IEEE Symposium on Security and Privacy (S&P 2006), 21-24 May 2006, Berkeley, California, USA. pp. 140–154. IEEE Computer Society (2006)

[19] Blanchet, B., Abadi, M., Fournet, C.: Automated verification of selected equivalences for security protocols. Journal of Logic and Algebraic Programming 75(1), 3–51 (Feb–Mar 2008)

[20] Blanchet, B., Smyth, B.: Automated reasoning for equivalences in the applied pi calculus with barriers. In: Computer Security Foundations Symposium (CSF), 2016 IEEE 29th. pp. 310–324. IEEE (2016)

[21] Blanchet, B., Smyth, B., Cheval, V.: Automatic Cryptographic Protocol Verifier, User Manual and Tutorial (2016)

[22] Brusó, M.: Dissecting unlinkability. Ph.D. thesis, Ph. D. dissertation, Technische Universiteit Eindhoven (2014)

[23] Brusó, M., Chatzikokolakis, K., Etalle, S., Den Hartog, J.: Linking unlinkability. In: International Symposium on Trustworthy Global Computing. pp. 129–144. Springer (2012)

[24] Chadha, R., Cheval, V., Ştefan Ciobâcă, Kremer, S.: Automated verification of equivalence properties of cryptographic protocols. ACM Trans. Comput. Log. 17(4), 23:1–23:32 (2016)

[25] Chaum, D.: Blind signatures for untraceable payments. In: Advances in Cryptology: Proceedings of CRYPTO '82. pp. 199–203. Plenum Press (1982)

[26] Chaum, D., Fiat, A., Naor, M.: Untraceable electronic cash. In: Advances in Cryptology: Proceedings of CRYPTO '88. LNCS, vol. 403, pp. 319–327. Springer (1990)

[27] Cheval, V., Comon-Lundh, H., Delaune, S.: Trace equivalence decision: Negative tests and non-determinism. In: 18th Conference on Computer and Communications Security (CCS'11). ACM, Chicago, Illinois, USA (Oct 2011)

[28] Chevalier, Y., Küsters, R., Rusinowitch, M., Turuani, M.: An NP decision procedure for protocol insecurity with XOR. In: LICS. pp. 261–270. IEEE Computer Society (2003)

[29] Chien, H.Y., Huang, C.W.: A lightweight RFID protocol using substring. In: Embedded and Ubiquitous Computing (EUC). pp. 422–431 (2007)

[30] Comon-Lundh, H., Delaune, S.: The finite variant property: How to get rid of some algebraic properties. In: Giesl, J. (ed.) Term Rewriting and Applications, 16th International Conference, RTA. LNCS, vol. 3467, pp. 294–307. Springer (2005)

[31] Cremers, C.J.F.: The Scyther Tool: Verification, falsification, and analysis of security protocols. In: 20th International Conference on Computer Aided Verification (CAV'08). LNCS, vol. 5123, pp. 414–418. Springer (2008)

[32] Delaune, S., Ryan, M.D., Smyth, B.: Automatic verification of privacy properties in the applied pi-calculus. In: Proceedings 2nd Joint iTrust and PST Conferences on Privacy, Trust Management and Security, (IFIPTM'08). Springer (2008)

[33] Deursen, T.v., Radomirović, S.: Algebraic attacks on RFID protocols. In: Information Security Theory and Practices. Smart Devices, Pervasive Systems, and Ubiquitous Networks (WISTP'09). Lecture Notes in Computer Science, vol. 5746, pp. 38–51. Springer (2009)

[34] Deursen, T.v., Radomirović, S.: Attacks on RFID protocols (version 1.1). Cryptology ePrint Archive, Report 2008/310 (August 2009), http://eprint.iacr.org/2008/310

[35] Dreier, J., Duménil, C., Kremer, S., Sasse, R.: Beyond subterm-convergent equational theories in automated verification of stateful protocols. In: Principles of Security and Trust - 6th International Conference, POST 2017. LNCS, vol. 10204, pp. 117–140. Springer (2017)

[36] Dreier, J., Kassem, A., Lafourcade, P.: Formal analysis of e-cash protocols. In: SECRYPT 2015 - Proceedings of the 12th International Conference on Security and Cryptography. pp. 65–75. SciTePress (2015)

[37] Escobar, S., Meadows, C., Meseguer, J.: Maude-NPA: Cryptographic protocol analysis modulo equational properties. In: Foundations of Security Analysis and Design V. LNCS, vol. 5705, pp. 1–50. Springer (2009)

[38] Escobar, S., Sasse, R., Meseguer, J.: Folding variant narrowing and optimal variant termination. Journal of Logic and Algebraic Programming 81(7-8), 898–928 (2012)

[39] Guttman, J.D., Ramsdell, J.D.: CPSA: a cryptographic protocol shapes analyzer (2009), http://hackage.haskell.org/package/cpsa

[40] Hirschi, L., Baelde, D., Delaune, S.: A method for verifying privacy-type properties: The unbounded case. In: IEEE Symposium on Security and Privacy, SP 2016. pp. 564–581. IEEE Computer Society (2016)

[41] Hirschi, L., Baelde, D., Delaune, S.: A method for unbounded verification of privacy-type properties. CoRR abs/1710.02049 (2017), http://arxiv.org/abs/1710.02049

[42] Kim, I.J., Choi, E.Y., Lee, D.H.: Secure mobile RFID system against privacy and security problems. In: SecPerU 2007 (2007)

[43] Kremer, S., Künnemann, R.: Automated analysis of security protocols with global state. Journal of Computer Security 24(5), 583–616 (2016)

[44] Küsters, R., Truderung, T.: Reducing protocol analysis with XOR to the xor-free case in the horn theory based approach. In: Proceedings of the 2008 ACM Conference on Computer and Communications Security, CCS 2008. pp. 129–138. ACM (2008)

[45] Lee, S., Asano, T., Kim, K.: RFID mutual authentication scheme based on synchronized secret information. In: Symposium on Cryptography and Information Security. Hiroshima, Japan (January 2006)

[46] Lowe, G.: Breaking and fixing the Needham-Schroeder public-key protocol using FDR. In: TACAS. pp. 147–166 (1996)

[47] Lowe, G.: A hierarchy of authentication specifications. In: 10th Computer Security Foundations Workshop (CSFW '97), June 10-12, 1997, Rockport, Massachusetts, USA. pp. 31–44. IEEE Computer Society (1997)

[48] Meier, S., Schmidt, B., Cremers, C.J.F., Basin, D.: The TAMARIN Prover for the Symbolic Analysis of Security Protocols. In: CAV. LNCS, vol. 8044, pp. 696–701. Springer (2013)

[49] O'Hanlon, P., Borgaonkar, R., Hirschi, L.: Mobile subscriber wifi privacy. In: Proceedings of Mobile Security Technologies (MoST'17), held as part of the IEEE Computer Society Security and Privacy Workshops. San Jose, CA, USA (May 2017)
[50] Ramsdell, J.D., Dougherty, D.J., Guttman, J.D., Rowe, P.D.: A hybrid analysis for security protocols with state. In: Proc. 11th International Conference on Integrated Formal Methods (IFM'14). LNCS, vol. 8739, pp. 272–287. Springer (2014)
[51] Santiago, S., Escobar, S., Meadows, C., Meseguer, J.: A formal definition of protocol indistinguishability and its verification using Maude-NPA. In: Security and Trust Management (STM) 2014. pp. 162–177. Springer (2014)
[52] Sasse, R., Escobar, S., Meadows, C., Meseguer, J.: Protocol analysis modulo combination of theories: A case study in Maude-NPA. In: STM. pp. 163–178 (2010)
[53] Schmidt, B.: Formal Analysis of Key Exchange Protocols and Physical Protocols. PhD dissertation, ETH Zurich (2012)
[54] Schmidt, B., Meier, S., Cremers, C.J.F., Basin, D.: Automated analysis of Diffie-Hellman protocols and advanced security properties. In: Computer Security Foundations Symposium (CSF). pp. 78–94. IEEE (2012)
[55] Tamarin repository. https://github.com/tamarin-prover/tamarin-prover (2018), accessed: 2018-04-27
[56] Tamarin website. https://tamarin-prover.github.io/ (2018), accessed: 2018-04-27
[57] Turuani, M.: The cl-atse protocol analyser. In: Term Rewriting and Applications, 17th International Conference, RTA 2006. LNCS, vol. 4098, pp. 277–286. Springer (2006)
[58] Unruh, D.: The impossibility of computationally sound xor. IACR Cryptology ePrint Archive 2010, 389 (2010)

APPENDIX

In the following, we shall deal with dependency graphs modulo *AC'* for *ALL*.

A. Existing Normal Form Conditions

We integrate the concept of normal message deduction, with construction and deconstruction rules, to dependency graphs. In this appendix we use an additional layer of ordering on K facts, and give the $\downarrow$ a subscript d, respectively e, to annotate whether it was already used in a bilinear pairing, which is only allowed once.

We adapt the previous normal form conditions [35] and add the new **N13** condition.

Definition 9 (Adapted from [35])**.** A *pre-normal dependency graph* for a set of protocol rules P is a dependency graph dg such that $dg \in dgraphs_{AC'}(\lceil P \rceil^{ALL}_{insts} \cup ND)$ and the following conditions are satisfied:

N1. The dependency graph dg is $\downarrow_{ALL}$-normal.

N2. There is no multiplication rule that has a premise fact of the form $K^{\uparrow}(s * t)$ and all conclusion facts of the form $K^{x}(s * t)$ are conclusions of a multiplication rule.

N3. If there are two conclusions c and c' with conclusion facts $K^{x}(m)$ and $K^{x'}(m')$ such that $m =_{AC'} m'$ and either $x = x' = \uparrow$, or $x = \downarrow y$ and $x' = \downarrow y'$ for $y,\ y' \in \{d, e\}$, then $c = c'$.

N4. All conclusion facts $K^{\uparrow}(f(t_1, \ldots, t_n))$ where f is an invertible function symbol are conclusions of the construction rule for f.

N5. If a node i has a conclusion $K^{\downarrow y}(m)$ for $y \in \{d, e\}$ and a node j has a conclusion $K^{\uparrow}(m')$ with $m =_{AC'} m'$, then $i < j$ and either $root(m)$ is invertible or the node j is an instance of coerce.

N6. There is no node $[K^{\downarrow d}(a),\ K^{\uparrow}(b)] \text{-}[]\!\!\rightarrow K^{\downarrow e}(c\hat{}r)$ where c does not contain any fresh names and $nifactors(r) \subseteq_{AC'} nifactors(b)$.

N7. There is no construction rule for $\sharp$ that has a premise of the form $K^{\uparrow}(s \sharp t)$ and all conclusion facts of the form $K^{\uparrow}(s \sharp t)$ are conclusions of a construction rule for $\sharp$.

N8. The conclusion of a deconstruction rule for $\sharp$ is never of the form $K^{\downarrow d}(s \sharp t)$.

N9. There is no node $[K^{\downarrow d}(a), K^{\uparrow}(b)] - [] \rightarrow K^{\downarrow e}([d]c)$ such that c does not contain any fresh names and $nifactors(d) \subseteq_{AC'} nifactors(b)$.

N10. There is no node i labeled with $[K^{\downarrow d}([t_1]p), K^{\downarrow d}([t_2]q)] - [] \rightarrow K^{\downarrow d}(\hat{e}(p, q)\hat{}c)$ such that there is a node j labeled with $[K^{\downarrow d}(\hat{e}(p, q)\hat{}c), K^{\uparrow}(r)] - [] \rightarrow K^{\downarrow d}(\hat{e}(p, q)\hat{}s)$, an edge $(i, 1) \rightarrowtail (j, 1)$, $nifactors(ti) \subseteq_{AC'} nifactors(r)$ for $i = 1$ or $i = 2$, and $\hat{e}(p, q)$ does not contain any fresh names.

N11. There is no node $[K^{\downarrow d}([a]p), K^{\downarrow d}([b]q)] - [] \rightarrow K^{\downarrow d}(\hat{e}(p, q)\hat{}(a * b))$ such that the send-nodes of the first and second premises are labeled with ru_1 and ru_2 and $fsyms(ru_2) <_{fs} fsyms(ru_1)$ where $<_{fs}$ is a total order on sequences of fact symbols.

N12. There is no chain of nodes repeatedly instantiating a rule of the form $\mathsf{K}^{\downarrow}(l|_p), \mathsf{K}^{\uparrow}(t_1), \ldots, \mathsf{K}^{\uparrow}(t_i) \text{-}[]\!\!\rightarrow \mathsf{K}^{\downarrow}(r)$ of length at least equal to the number of subterms of $l|_p$, if $l|_p$ and r are unifiable.

dg is a *normal dependency graph* if it additionally satisfies the following condition:

N13. There is no chain of repeated instantiations of the deconstruction rules for XOR.

We denote the set of all pre-normal dependency graphs (resp. normal dependency graphs) for P by $pdgraphs(P)$ (resp. $ndgraphs(P)$).

Most of the conditions avoid redundancy and enforce that all terms are in normal form (**N1**). **N2** and **N6** are for the $\mathcal{DH}$ theory, **N7** to **N11** are for $\mathcal{BP}$. **N3** avoids multiple deduction of the same term. **N4** forbids deduction by coerce rule on invertible function symbols that instead must be created by construction. **N5** enforces the ordering of $\mathsf{K}^{\downarrow}$ before $\mathsf{K}^{\uparrow}$ facts. **N12** limits the length of derivation chains for certain rules to the maximum required number. For detailed explanations of these normal form conditions, see [54], [53], [35].

B. Proof of Theorem 1 for XOR.

We define some sets of messages, denoting notions of adversary knowledge over a dependency graph.

Definition 10. We define the $\mathtt{known}$ messages of a dependency graph dg as well as the construction-, deconstruction- and combined-known messages for (pre-)normal dependency graphs ndg:

$$
\begin{aligned}
\mathtt{known}(dg) &= \{m \mid \text{exists conclusion } \mathsf{K}(m) \in dg\} \\
\mathtt{known}^{\uparrow}(ndg) &= \{m \mid \text{exists conclusion } \mathsf{K}^{\uparrow}(m) \in ndg\} \\
\mathtt{known}^{\downarrow}(ndg) &= \{m \mid \text{exists conclusion } \mathsf{K}^{\downarrow}(m) \in ndg\} \\
\mathtt{known}^{\updownarrow}(ndg) &= \mathtt{known}^{\uparrow}(ndg) \cup \mathtt{known}^{\downarrow}(ndg)
\end{aligned}
$$

We define as $\mathtt{stfacts}(dg)$ all state conclusion facts of a (pre-)normal dependency graph except for adversary knowledge facts $\mathsf{K}, \mathsf{K}^{\uparrow}, \mathsf{K}^{\downarrow}$. We define as the created messages $\mathtt{created}(dg)$ all the fresh values that appear under Fr facts in a (pre-)normal dependency graph.

Finally, we say that a (pre-)normal dependency graph $ndg' = (I', D')$ is a deduction extension of $ndg = (I, D)$ if I is a prefix of I', $D \subseteq D'$, $trace(ndg) = trace(ndg')$, $\mathtt{stfacts}(ndg) = \mathtt{stfacts}(ndg')$, and $\mathtt{created}(ndg) = \mathtt{created}(ndg')$.

Note that if $t \in \mathtt{known}^{\updownarrow}(ndg)$, then there exists a deduction extension ndg' of ndg such that $t \in \mathtt{known}^{\uparrow}(ndg')$ using an instance of the COERCE rule if necessary.

Definition 11 (Factors)**.** We define the *factors* of a term t as follows:

$$
factors(t) = \begin{cases} \{t_1, \ldots, t_n\} & \text{if } t =_{AC'} t_1 \oplus \ldots \oplus t_n \\ \varnothing & \text{otherwise} \end{cases}
$$

Note that, $\oplus$ is a binary function symbol and *AC'* only modifies the order of the arguments of XOR, but never removes an argument. Therefore, $factors(t)$ is never a singleton set.

We can now show an intermediate lemma required for the proof of the correctness theorem, Theorem 1. The lemma simply states that if two terms are known by the adversary, he can also compute XOR of both terms while complying with all normal form conditions except **N13** which will be treated directly in the proof of Theorem 1.

Lemma 3. *For all pre-normal dependency graphs ndg with $t_1 \in_{AC'} \mathtt{known}^{\updownarrow}(ndg)$ and $t_2 \in_{AC'} \mathtt{known}^{\updownarrow}(ndg)$ there exists a deduction extension ndg' of ndg such that $(t_1 \oplus t_2 \downarrow_{ALL}) \in_{AC'} \mathtt{known}^{\updownarrow}(ndg')$.*

Proof. If $(t_1 \oplus t_2 \downarrow_{ALL}) \in_{AC'} \mathtt{known}^{\updownarrow}(ndg)$ then we trivially conclude with $ndg' = ndg$. In the following, we thus assume that $(t_1 \oplus t_2 \downarrow_{ALL}) \notin_{AC'} \mathtt{known}^{\updownarrow}(ndg')$.

If $t_1 =_{AC'} t_2$, then $t_1 \oplus t_2 \downarrow_{ALL} 0$, and we extend ndg with an instance of the constructor rule for 0. We now assume that $t_1 \neq_{AC'} t_2$.

Let $f_i = factors(t_i)$ and $n_i = |f_i|$ for $1 \leq i \leq 2$. We proceed by induction on $k = n_1 + n_2$.

Base Case. If $k = 0$, then we have $n_1 = n_2 = 0$, i.e., $f_1 = f_2 = \varnothing$. Therefore, the facts $\mathsf{K}^x(t_1)$ and $\mathsf{K}^y(t_2)$ given by hypothesis were not conclusions of some variant of a XOR rule instance. We also have that $t_1 \oplus t_2 \downarrow_{ALL} =_{AC'} t_1 \oplus t_2$. Indeed, t_1 and t_2 are different values in normal forms, have no $\oplus$ at top level and other symbols do not interact with XOR ($\oplus$ neither appears in *DHBPM* nor in *ACC*). Potentially using some Coerce rules (depending on x and y), we can extend ndg to produce facts $\mathsf{K}^{\uparrow}(t_1)$ and $\mathsf{K}^{\uparrow}(t_2)$. We extend again the graph with an instance of the constructor rule for $\oplus$ with $\mathsf{K}^{\uparrow}(t_1)$ and $\mathsf{K}^{\uparrow}(t_2)$ as premises and $\mathsf{K}^{\uparrow}(t_1 \oplus t_2)$ as conclusion. All normal form conditions are satisfied. Note that **N3** holds by assumption (i.e., $(t_1 \oplus t_2 \downarrow_{ALL}) \notin_{AC'} \mathtt{known}^{\updownarrow}(ndg')$) and **N5** holds because ndg does not violate it.

Inductive Case. Otherwise, $k > 0$. There must be some i such that $n_i \geq 2$. We suppose w.l.o.g. $n_1 \geq n_2$, hence $n_1 \geq 2$ and $f_1 \neq \varnothing$.

- If $f_1 \cap f_2 = \varnothing$ then $t_1 \oplus t_2 \downarrow_{ALL} =_{AC'} t_1 \oplus t_2$ (since t_1 and t_2 are normal forms). We extend ndg with an instance of the constructor rule for $\oplus$ with $\mathsf{K}^{\uparrow}(t_1)$ and $\mathsf{K}^{\uparrow}(t_2)$ as premises and $\mathsf{K}^{\uparrow}(t_1 \oplus t_2)$ as conclusion. We assume in the following that $f_1 \cap f_2 \neq \varnothing$. Note that $f_1 \not\subseteq f_2$ since $f_1 \neq f_2$ and $n_1 \geq n_2$.
- If $f_2 \not\subseteq f_1$.
  - If $t_1 \in_{AC'} \mathtt{known}^{\downarrow}(ndg)$, we can use an instance of the second deconstruction rule with $\mathsf{K}^{\downarrow}(t_1)$ and $\mathsf{K}^{\uparrow}(t_2)$ as premises and $\mathsf{K}^{\downarrow}(t_1 \oplus t_2 \downarrow_{ALL})$ as conclusion. In case $t_2 \notin_{AC'} \mathtt{known}^{\uparrow}(ndg)$, then $t_2 \in_{AC'} \mathtt{known}^{\downarrow}(ndg)$ and one can use a Coerce rule to add $\mathsf{K}^{\uparrow}(t_2)$ to the graph.
  - If $t_2 \in_{AC'} \mathtt{known}^{\downarrow}(ndg)$, we can use an instance of the second deconstruction rule with $\mathsf{K}^{\downarrow}(t_2)$ and $\mathsf{K}^{\uparrow}(t_1)$ as premises and $\mathsf{K}^{\downarrow}(t_1 \oplus t_2 \downarrow_{ALL})$ as conclusion. In case $t_1 \notin_{AC'} \mathtt{known}^{\uparrow}(ndg)$, then $t_1 \in_{AC'} \mathtt{known}^{\downarrow}(ndg)$ and one can use a Coerce rule to add $\mathsf{K}^{\uparrow}(t_1)$ to the graph.
  - Otherwise, it holds that $t_1 \notin_{AC'} \mathtt{known}^{\downarrow}(ndg)$ and $t_2 \notin_{AC'} \mathtt{known}^{\downarrow}(ndg)$. Therefore, we have $t_1 \in_{AC'} \mathtt{known}^{\uparrow}(ndg)$ and $t_2 \in_{AC'} \mathtt{known}^{\uparrow}(ndg)$, and both t_1 and t_2 must be the conclusion of a constructor rule instance. Let r_1 and r_2 be the premises of the constructor rule instance for t_1, and s_1 and s_2 be the premises of the constructor rule instance for t_2, then we have that $factors(r_1) \cap factors(r_2) = \varnothing$ and $factors(s_1) \cap factors(s_2) = \varnothing$. Indeed, otherwise, the application of the constructor on r_1 and r_2 (or s_1 and s_2, respectively) would not be a valid normal form instantiation of the constructor rule (i.e., $r_1 \oplus r_2$ or $s_1 \oplus s_2$, respectively, would not be $\downarrow_{ALL}$-normal). As $f_1 \cap f_2 \neq \varnothing$, there exists i and j such that $factors(r_i) \cap factors(s_j) \neq \varnothing$. By the induction hypothesis there exists a deduction extension ndg' such that $r_i \oplus s_j \in_{AC'} \mathtt{known}^{\updownarrow}(ndg')$. We have that $|factors(r_i \oplus s_j)| \leq |factors(r_i)| + |factors(s_j)| - 1$, hence by applying the inductive hypothesis with r_{3-i} and then with s_{3-j}, we obtain a deduction extension ndg'' of ndg' such that $r_1 \oplus r_2 \oplus s_1 \oplus s_2 =_{E_{ALL}} t_1 \oplus t_2 \in_{AC'} \mathtt{known}^{\updownarrow}(ndg'')$.
- If $f_2 \subseteq f_1$.
  - If $t_1 \in_{AC'} \mathtt{known}^{\downarrow}(ndg)$, we can use an instance of the first deconstruction rule with $\mathsf{K}^{\downarrow}(t_1)$ and $\mathsf{K}^{\uparrow}(t_2)$ as premises and $\mathsf{K}^{\downarrow}(t_1 \oplus t_2 \downarrow_{ALL})$ as conclusion. In case $t_2 \notin_{AC'} \mathtt{known}^{\uparrow}(ndg)$, then $t_2 \in_{AC'} \mathtt{known}^{\downarrow}(ndg)$ and one can use a Coerce rule to add $\mathsf{K}^{\uparrow}(t_2)$ to the graph.
  - If $t_1 \notin_{AC'} \mathtt{known}^{\downarrow}(ndg)$, then t_1 must be the result of a constructor rule. Let s_1 and s_2 be the premises of this constructor rule. It holds that $factors(s_1) \cap factors(s_2) = \varnothing$. Indeed, otherwise, the application of the constructor would not be a valid normal form instantiation of the constructor rule (i.e., $s_1 \oplus s_2$ would not be $\downarrow_{ALL}$-normal). Since $f_1 \cap f_2 \neq \varnothing$, there exists i such that $factors(s_i) \cap factors(t_2) \neq \varnothing$. By induction hypothesis, there exists a deduction extension ndg' such that $s_i \oplus t_2 \in_{E_{ALL}} \mathtt{known}^{\updownarrow}(ndg')$. We have that $|factors(s_i \oplus t_2)| \leq n_2 + |factors(s_i)| - 1$, hence there exists a deduction extension ndg'' of ndg' such that $s_{3-i} \oplus s_i \oplus t_2 =_{E_{ALL}} t_1 \oplus t_2 \in_{AC'} \mathtt{known}^{\updownarrow}(ndg'')$.

□

$$1:\ \frac{\mathsf{K}^{\downarrow}(t_1)\quad \mathsf{K}^{\uparrow}(t_2)}{\mathsf{K}^{\downarrow}(t_3)} \qquad 2:\ \frac{\mathsf{K}^{\downarrow}(t_3)\quad \mathsf{K}^{\uparrow}(t_4)}{\mathsf{K}^{\downarrow}(t_5)} \qquad \longrightarrow \qquad 1:\ \frac{\text{(by Lemma 3)}}{\mathsf{K}^{\uparrow}(t_2 \oplus t_4)} \qquad 2:\ \frac{\mathsf{K}^{\downarrow}(t_1)\quad \mathsf{K}^{\uparrow}(t_2 \oplus t_4)}{\mathsf{K}^{\downarrow}(t_5)}$$

Fig. 13: Merging multiple deconstruction rule instances.

N13: $\Gamma \rightsquigarrow_P \bot$
if $i : ri, j : ri', (i,1) \dashrightarrow (j,1)$, ri and ri' are instances of XOR deconstruction rules

Fig. 14: New constraint-reduction rule.

We can now prove Theorem 1 by adapting the proof we find in [35]. The proof considers a dependency graph on which we add a rule instance and see if it is convertible into a possible rule instance from $ginst(\lceil P \cup ND \rceil^{ALL}_{insts} \cup \{Fresh\})$ to complete an equivalent normal dependency graph.

Proof of Theorem 1. Relying on Lemmas 1, 2, 3 we shall prove that

$$\overline{trace(\{dg \mid dg \in dgraphs_{AC'}(\lceil P \cup MD \rceil^{E_{ALL}}) \wedge dg \downarrow_{ALL} - normal\})} =_{AC'} \overline{trace(ndgraphs(P))}$$

We prove both inclusions of the above by structural induction on (pre-) normal dependency graphs. Most cases can be treated as in [35].

$\subseteq_{AC'}$: We prove that, for all $dg \in dgraphs_{AC'}(\lceil P \cup MD \rceil^{ALL}_{insts})$ with $dg \downarrow_{ALL}$ -normal, there is $ndg \in ndgraphs(P)$ such that:

$$known(dg) \subseteq_{AC'} \mathtt{known}^{\updownarrow}(ndg) \quad (1)$$
$$stfacts(dg) \subseteq_{AC'} stfacts(ndg) \quad (2)$$
$$created(dg) =_{AC'} created(ndg) \quad (3)$$
$$\overline{trace(dg)} =_{AC'} \overline{trace(ndg)} \quad (4)$$

Notice that the inclusion of (2) applies to multisets.

The four properties hold for $dg = ([], \varnothing)$. Let $dg = (I, D) \in dgraphs_{AC'}(\lceil P \cup MD \rceil^{ALL}_{insts})$ with $dg \downarrow_{ALL}$ -normal, and $ndg = (\tilde{I}, \tilde{D}) \in ndgraphs(P)$ such that (1)-(4) hold. Let $ri \in ginst_{AC'}(\lceil P \cup MD \rceil^{ALL}_{insts} \cup \{Fresh\})$ such that $dg' = (I \cdot ri, D \uplus D') \in dgraphs_{AC'}(\lceil P \cup MD \rceil^{ALL}_{insts})$. We must show that there is a deduction extension ndg' of ndg that satisfies (1)-(4) with respect to dg'.

We perform a case distinction on ri:

- for $ri \in ginst_{AC'}(\lceil P \rceil^{ALL}_{insts} \cup \{Fresh\})$, we can extend ndg to $ndg' = (\tilde{I} \cdot ri, \tilde{D} \uplus \tilde{D}')$ in accordance with conditions (2) and (3),
- for $ri \in \{Out(m) \dashrightarrow [] \rightarrow K(m), K(m) \dashrightarrow [K(m)] \rightarrow In(m), Fr(n) \dashrightarrow [] \rightarrow K(n), [] \rightarrow K(c)\} \subset ginst_{ACC}(\lceil MD \rceil^{\mathcal{R}_{CST'}}_{insts})$, there are corresponding rules in ND that can complete $\tilde{I}$,
- for all variants of the XOR rule we can apply Lemma 3. In case the dependency graph violates **N13**, we have two consecutive instances of XOR deconstruction rules, which we can replace with one instance, as illustrated in Figure 13. By Lemma 3, there exists a deduction extension to obtain $\mathsf{K}^{\uparrow}(t_2 \oplus t_4)$. Note that although the application of Lemma 3 can again introduce new instances that violate **N13**, we can apply the same technique again, which will terminate as there is only a finite number of known terms, and the application of the lemma does not duplicate terms.
- Other rules only deal with the bilinear-pairing, Diffie-Hellman or user-defined theory, and the addition of XOR does not interfere with the original proof.

$\supseteq_{AC'}$: For the other inclusion, we show, by an analogous way, that for all $ndg \in ndgraphs(P)$, there is a $dg \in dgraphs_{AC'}(\lceil P \cup MD \rceil^{ALL}_{insts})$ with $dg \downarrow_{ALL}$ -normal, such that:

$$known(ndg) \subseteq_{AC'} \mathtt{known}^{\updownarrow}(dg) \quad (1)$$
$$stfacts(ndg) \subseteq_{AC'} stfacts(dg) \quad (2)$$
$$created(ndg) =_{AC'} created(dg) \quad (3)$$
$$\overline{trace(ndg)} =_{AC'} \overline{trace(dg)} \quad (4)$$

It holds for $ndg = ([], \varnothing)$. Let $ndg = (I, D) \in ndgraphs(P)$, and $dg = (\tilde{I}, \tilde{D}) \in dgraphs_{AC'}(\lceil P \cup MD \rceil^{ALL}_{insts})$ with $dg \downarrow_{ALL}$ -normal such that (1)-(4) hold. Let $ri \in ginst_{AC'}(\lceil P \rceil^{ALL}_{insts} \cup ND \cup \{Fresh\})$ such that $ndg' = (I \cdot ri, D \uplus D') \in ndgraphs_{AC'}(P)$. We must show that there is a deduction extension dg' of dg that satisfies (1)-(4) with respect to ndg'. Note that there is no normal dependency graph violating **N12** by definition, so no special care needs to be taken for that restriction in this direction. We still perform a case distinction on ri

- for $ri \in ginst_{AC'}(\lceil P \rceil^{ALL}_{insts} \cup \{Fresh\})$: analogous to the other inclusion,
- for $ri \in ginst_{AC'}(ND)$: there is a corresponding rule in $\lceil MD \rceil^{ALL}_{insts}$ for all variants of XOR rule instances ri. For the others rules, we proceed as in the original proof.

□

C. Proof of Soundness of Equivalence Mode for XOR.

In [14] it is shown that dependency graph equivalence (TAMARIN's version of diff-equivalence) is a sound, but incomplete approximation of observational equivalence for multiset rewriting rules [14, Theorem 1]. This proof holds for any set of protocol and adversary rules, and thus also applies in our case. Then, in a second step, in the appendix of [14] it is shown that it is sufficient to consider only normalized graphs when checking for dependency graph equivalence [14, Theorem 3]. This result carries over to our case, there is only a syntactic change from *AC* to *AC'* as we need to include XOR in the set of *AC* symbols, but this does not change the rest of the proof.

We then only need to show soundness and completeness of our new normal form condition **N13**, i.e., to show that a bi-system is dependency graph equivalent if, and only if, it is dependency graph equivalent when only considering graphs respecting **N13**.

Theorem 2 (Soundness and Completeness of N13 w.r.t. Dependency Graph Equivalence)**.** *Let S be a $*$-restricted protocol bi-system. S is dependency graph equivalent if, and only*

*if, it is dependency graph equivalent when only considering graphs respecting **N13**.*

Proof. One direction ($\Rightarrow$) is trivial: each graph ensuring **N13** is a valid graph, hence by assumption there is a valid mirror that also ensures **N13** as this property is preserved by mirroring.

For the other direction ($\Leftarrow$), we need to show that even graphs that violate **N13** have valid mirrors, assuming that all graphs ensuring **N13** have a valid mirror.

Assume that there is a dependency graph dg that does violate **N13**. Using the same approach as in the proof of Theorem 1 illustrated in Figure 13, based on Lemma 3, we can turn dg into a graph dg' that ensures **N13**.

By assumption there are the necessary valid mirrors for dg'. We then need to show that there are also the necessary valid mirrors for dg. The only difference between dg and dg' is part of the adversary deduction, where the adversary applied XOR deconstruction or construction rules. One can see the modifications as computing the same result, but applying the XOR operations on the terms in a different order. Because XOR is associative and commutative, the result remains the same. We can thus "undo" the modifications on the mirrors of dg', i.e., compute the result as in dg and not as in dg', and obtain valid mirrors for dg. This is possible in particular as we relaxed the mirroring to allow deconstruction rules to be mirrored using construction rules, and vice versa. □

D. Modified Constraint-Solving Rules

We only added one constraint-solving rule that implements **N13**, as shown in Figure 14. The proof of soundness and completeness of the constraint solving is then a straightforward extension of the proofs of [14, Lemma 6] and [48, Theorem 2]: It is easy to see that this rule does not create any new solutions, and it only removes solutions violating **N13** as we only remove systems containing two chained instances of deconstruction rules for XOR.